COMPUTATIONAL FLUID DYNAMICS IN AN EQUATION-BASED, ACAUSAL MODELING ENVIRONMENT

A Dissertation
Presented to
The Academic Faculty

by

Jason Brown

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Architecture

Georgia Institute of Technology
December, 2010

COMPUTATIONAL FLUID DYNAMICS IN AN EQUATION-BASED, ACAUSAL MODELING ENVIRONMENT

Approved by:

Godfried Augenbroe, Advisor
School of Architecture
Georgia Institute of Technology

Dr. Ruchi Choudhary
Civil and Environmental Engineering Division
University of Cambridge

Dr. Russell Gentry
School of Architecture
Georgia Institute of Technology

Dr. Christiaan Paredis
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Michael Wetter
Simulation Research Group
Lawrence Berkeley National Laboratory

Date Approved: November 12, 2010

PREFACE

This work seeks to combine topics which are traditionally separate into one framework. In the course of this, two uses of the word 'domain' are encountered which have different meanings depending on the topic. One refers to areas of technical inquiry or analysis techniques; for example heat transfer is a domain separate from controls, or CFD simulations constitute a domain separate from multizone simulations. Within the topic of CFD, another use of the word domain appears: in this it refers to a region of space within which the CFD simulation is being conducted. While it should be clear from the context which meaning applies, this note is given to the reader as a pre-emptive clarification.

ACKNOWLEDGEMENTS

This work would not have been possible without the unconditional support and patience of my wonderful wife Rebeccah. I am constantly reminding myself how lucky I am to have found such an incredible person to share my life with.

I also wish to acknowledge the rich and rewarding encounters I have had while at the College of Architecture. Fried Augenbroe and John Peponis in particular struck me for the depth of their thinking, always considering the 'whys' as well as the 'whats'. Russell Gentry's kindness and nuts and bolts outlook have also helped to keep me grounded during my time here. I also wish to thank Ruchi Choudhary, Chris Paredis, and Michael Wetter, the remaining members of my thesis committee, for their support of my work and engaging conversations.

To my fellow students I wish to thank for their support. Huafen Hu, Jeannie Kim, and Matt Erwin saved me with their immense service during my teaching responsibilities. I could not have taught those classes without the many hours of work they put in. Thank you also to all the building technology graduate students for your kindness.

Finally I wish to thank the staff and faculty of the College of Architecture, who have supported me through these years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CFD: Computational fluid dynamics

DAE: Differential algebraic equations

ES: Energy simulation

RANS: Reynolds-averaged Navier-Stokes

LES: Large-eddy simulation

LIST OF SYMBOLS

Roman Symbols

$\boldsymbol{c}$ The peculiar velocity $\boldsymbol{\xi} - \boldsymbol{u}$

F_N Probability density function for the microstate of a system of N particles, all N particles inclusive

F_R Reduced probability density function for the microstate of a system of N particles, only R particles inclusive

f Scaled reduced probability density function, the single particle distribution $f = f_1 = NmF_1$

g Internal energy distribution

gm Modified internal energy distribution

h Convective heat transfer coefficient

$\tilde{h}$ Total energy distribution

k Boltzmann's constant

m Mass of a single particle

N Number of particles in a system

N_A Avogadro's number

n Number of replica systems in an ensemble, grid scaling factor

p Pressure

$\boldsymbol{p}$ Momentum vector

q Heat flux

$\boldsymbol{q}$ Heat flux vector

q_α Heat flux in direction α

Q Heat flow rate

R Specific ideal gas constant

S A surface in phase space

T Temperature

t Time or thickness

$\boldsymbol{u}$ Macroscopic velocity vector: $(u_x, u_y, u_z) = (u, v, w)$

w_i Weights of Gauss-Hermite quadrature

$\boldsymbol{x}$ Spatial coordinate: (x, y, z)

Greek Symbols

α General index, either for lattice directions or cartesian coordinate directions; thermal diffusivity

β General index, generally for lattice directions

Γ $6N$ dimensional phase space

γ General index; used for component direction x, y, or z

$\boldsymbol{\gamma}$ Coordinate in the $6N$ dimensional phase space

δ Small change, e.g. δT is a change in temperature

$\delta_{\alpha\beta}$ Kronecker delta

η Number of spatial degrees of freedom of a particle

θ Relaxation time

κ Thermal conductivity

λ Second viscosity coefficient

μ Dynamic (or absolute) viscosity

ν Kinematic viscosity

$\boldsymbol{\xi}$ Molecular (or microscopic) velocity vector

ξ_1 Velocity of particle (molecule) 1

$\boldsymbol{\xi}_i$ Molecular velocity vector of direction i

$\check{\xi}$ Lattice speed

$\boldsymbol{\varsigma}$ Combination vector of particle position vector and momentum vector

$\Phi_{\alpha\beta}$ Stress tensor; components are α and β

ρ Macroscopic density

τ Time constant; dimensionless relaxation time

τ_e Dimensionless relaxation time for the internal energy distributions

Υ Order of grid convergence

ψ Streamfunction

Ω_B The Boltzmann collision operator

ω Relaxation parameter in grid convergence computations

Subscripts

e Used in reference to energy

i Index of the molecular velocities in a discrete-molecular-velocity fluid system

k Index of the systems in an ensemble

N Number of particles in a system

r Number of particles considered in a reduced particle probability density function

α General index, either for lattice directions or cartesian coordinate directions

β General index, usually for lattice directions

SUMMARY

The practice of building simulation is split between domains such as energy, multizone airflow, computational fluid dynamics (CFD) airflow, and controls analysis, as well as between the tools which conduct these analyses. Previous work in the integration of these analyses and tools have focused on linking existing tools, written in algorithmic programming languages, together by interfacing them using coupling mechanisms implemented in algorithmic programming languages. This thesis takes a different approach, using the equation-based, object oriented modeling language Modelica to create models in different domains and interfaces between those models within a single framework which has benefits to the modeler/analyst in terms of both representation of physical processes and flexibility in modeling systems composed of many interacting components.

Specifically, the simulation of airflows within buildings has historically been compartmentalized into distinct domains such as nodal network (multizone) simulations and CFD. Such airflow simulations are also often treated independently of building energy simulations (via heat transfer) despite their interrelation. Recent work has reported on combining these types of analyses by linking pre-existing simulation software together. Here a prototype CFD package of models is built in Modelica and coupled to models of conductive heat transfer and controls. Comparisons of results of simulations so constituted to analytical solutions and benchmark data available in the literature show good agreement, indicating the technical viability of this approach. Limitations include the absence of turbulence modeling and the lack of modeling features which improve computational efficiency, such as non-uniform grids.

CHAPTER 1: INTRODUCTION

Simulation of the energy performance of buildings has contributed to numerous economic and infrastructural benefits in the decades since the original DOE-1 computer program and its descendant DOE-2 were released (Rosenfeld, 1999), and although building simulation has since grown into its own art and science, much of what has happened in the main is similar to what had already happened. However, what has been thought – outside the main – is different and has only gradually been happening on a small scale. This thinking seeks to i) relieve humans of much of the overhead of creating and using models, ii) synthesize domains which have historically been treated separately, and iii) in the process integrate different subcultures of the building simulation community such as energy and airflow simulation.

1.1 Multi-Domain Modeling in Building Performance Simulation

Buildings devote much of their energy budget towards the goal that the humans in them can function unburdened with thoughts of their environment, and one aspect of that environment is thermal comfort. The waste heat produced by a person needs to be removed from their body at the same rate it is produced, which is done by the three mechanisms of heat transfer: conduction, radiation, and convection. The latter two are the most significant with respect to thermal comfort, with convection being the mechanism most often employed to actively balance occupant heat transfer and create a thermally comfortable environment. Air at conditions suitable for this purpose is directed to occupants, exchanging energy and picking up contaminants such as CO_2. Air also contacts and exchanges energy with heat generators such as lights and surfaces like walls and windows. Suitable replacement air is introduced, either by passive or natural means or by mechanical conditioning.

Basic heat transfer and fluid mechanics, along with building systems such as lights, HVAC components, and control systems, are thus tightly coupled and all must be treated in building performance simulation. Despite this, and with some justification, there has traditionally been a split between tools whose primary function is to simulate the energy performance of a building – energy simulation (ES) programs like DOE-2 and EnergyPlus – and tools which focus on simulating the air flow within, through, and perhaps also around the building envelope. This split can exist because each tool makes assumptions about the domain which it does not focus on. In the case of EnergyPlus, airflows are assumed to be such that temperatures are uniform throughout the space – often a good assumption – and the convective heat transfer coefficient h is estimated from models (DOE, 2008).

There are situations when this is not the case however, and situations where the airflow needs to be quantified at a detail beyond what energy simulation programs can provide. For this there are three simulation approaches. The first is known by a variety of names – multizone, nodal, airflow network, *et al.* but referred to here as multizone – assumes fully mixed conditions, and postulates that flow paths through a building such as ducts or other mechanical systems, cracks in the facade, or open doors between rooms are linked together at nodes which represent rooms or other spaces. The flow along these flow paths is calculated from power law equations relating volume flow rates to pressure differences between the nodes. Chemical concentrations can also be tracked in addition to flow rates (Chen, 2009, Hensen, 2003 and Axley, 2007). In a recent survey, Chen (2009) found such methods to be accepted among practitioners for whole-building airflow analysis, although their market penetration may be hindered by the user-unfriendly interfaces of the most popular multizone analysis programs, CONTAM (Walton and Dols, 2008) and COMIS (Feustel, 1998).

The second method, the zonal approach, partitions a space such as a room into several – on the order of 10 – isothermal zones or cells, each with a possibly different

temperature. Conservation laws are applied in each cell to determine the flows of mass and energy between the cells. This method is therefore able to coarsely treat spaces with a nonuniform temperature distribution (Megri and Haghighat, 2007). Zonal method are intended as a balance between computing time and spatial resolution, although it has been argued (Chen, 2009) that their overall utility is comparable to coarse computational fluid dynamics (CFD) methods, the third approach.

In contrast to the reduced or simplified conservation laws used by the multizone and zonal approaches, CFD methods solve – or satisfy – the general mass, momentum, and energy balance laws and are the most sophisticated, detailed, and resource-intensive flow simulation techniques. As such they are reserved only for those cases and locations where their use is warranted: natural ventilation including wind- and buoyancy-driven flows, flows with high momentum, detailed studies on thermal comfort, etc. (Chen, 2009), as well as basic research (Addington, 2003). In professional practice CFD simulations are run using commercial packages such as FLUENT (ANSYS, 2010) or FloVENT (Mentor Graphics, 2010).

Thus, in addition to the split between energy simulation and airflow simulation, there is also this three-way split within airflow simulation. This latter split, however, exists for the good reason that there is no one-technique-fits-all approach to airflow simulation. Each technique has its niche. However, one simulation scenario may have more than one niche, and therefore there has been work on coupling these different airflow techniques. In some situations a multizone method may not give accurate results (Wang and Chen, 2008), and if such a situation occurs in one zone of an otherwise appropriate multizone model, then the use of a CFD model within a larger multizone model can provide a unique solution (Wang and Chen, 2007a) which improves the overall results (Wang and Chen, 2007b). Tan and Glicksman (2005) also coupled a multizone model to a CFD model and investigated the effect of choice of boundary between multizone and CFD volumes on the solution of a simulated

naturally ventilated space.

There has also been work in stitching the split between energy simulation (ES) and airflow modeling. Hensen (2003) gives a review, particularly in regard to coupling multizone models with energy simulation. The simulation environment TRNSYS incorporates a variety of methods to incorporate COMIS and CONTAM (Bradley and Kummërt, 2005), and likewise EnergyPlus has facilities for interfacing to COMIS (Crawley et al., 2001). Linking energy simulation to CFD has also received attention. The convective heat transfer coefficient h, a bridge between the two domains, can vary over several orders of magnitude (Lienhard(IV) and Lienhard(V), 2005) and is a large source of uncertainty, as is the rate of air infiltration into a building. In a study to determine the appropriateness of different energy/airflow simulation combinations, Djunaedy et al. (2004) found that depending on the design variations of a space, the uncertainty in h can cause up to a 66% deviation in the simulated maximum heating load, a 20% deviation in the simulated heating energy demand, and up to a 25% deviation in the simulated maximum cooling load. Corresponding deviations due to the uncertainty in the infiltration rate are 25%, 60%, 13%, respectively. CFD can be used to reduce these uncertainties since h can be computed as part of a CFD simulation, e.g. (Zhai and Chen, 2004), and likewise investigate the fluid-mechanical environment around a building to clarify the infiltration. The feasibility of coupling energy simulation to CFD has been demonstrated theoretically (Zhai and Chen, 2003) and in implementation (Zhai and Chen, 2005).

These efforts at stitching together separate domains of building performance simulation all seek to join otherwise independent simulation programs into a cooperative federation of simulation programs. Termed co-simulation or external coupling, this paradigm uses one of a variety of coupling methods (which physical variables manifest the coupling) and coupling strategies (how tightly are the two programs are linked, i.e. how consistent the two programs' solutions are to each other). These topics will

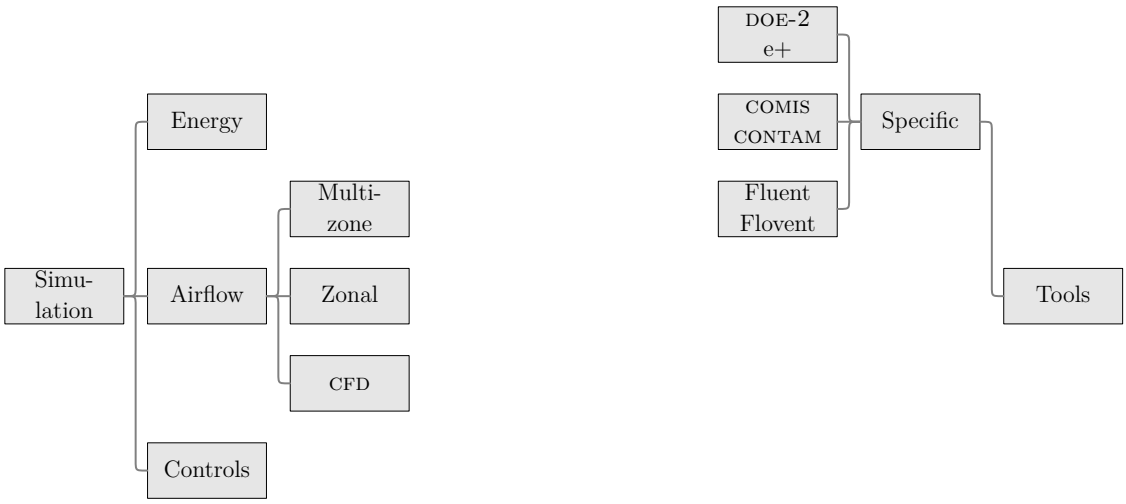


Figure 1.1.1 The Simulation Landscape with Associated Specific Tools

be expanded on in Chapter 2.

1.2 From Tools to Platforms for Tools

DOE-2 has been superseded by EnergyPlus, a significant evolutionary improvement with a more modular architecture, more sophisticated techniques, expanded capabilities, and provisions for the extension of the program including the creation of links to tools in domains such as lighting or nodal network airflow analysis, which are relevant but traditionally exist in their own self-contained realms (Crawley et al., 2001). One can infer from this last characteristic that flexibility in building simulation tools is a desired feature. Each building is almost always a unique artifact involving many energy and mass transfer processes that span multiple domains and occur within and across many different components of the building. It is unsurprising that flexibility and multi-domain capability would be high on the wish list.

Despite this improvement in modularity and extensibility, EnergyPlus remains a monolithic program: it is 'hard-wired' at its core to be application- and somewhat domain-specific, i.e. specific to building energy performance simulation. A much more fundamentally modular and extensible approach was taken by the developers of the Transient System Simulation Tool (TRNSYS). Roughly contemporaneous with the original DOE-1 and DOE-2 efforts and likewise growing out of a need to simulate the energy performance of buildings, TRNSYS aimed at the outset to be general and flexible by employing a component based modeling philosophy in which a model of a system is composed of independent and reusable individual component models that are connected together to form a larger model. The business of the program core is to simulate any model thus constructed (Bradley and Kummërt, 2005). Thus, though often used for buildings, TRNSYS may be used for the dynamic simulation of any system provided that the appropriate component models are available or could be developed. This flexibility and extensibility enables modelers to construct suitable models themselves instead of resorting to 'creative' *ad hoc* modeling hacks to trick a monolithic program into doing something that the program's developers did not and could not foresee being required. What makes TRNSYS flexible is that the developer's intent, if any, with respect to the program's intended area of application is irrelevant. The area of application is under the modeler's aegis, not the developer's. This flexibility allows the simulation of a building's thermal behavior simultaneously with the simulation of its systems (e.g. HVAC), something DOE-2 could not do and thus was a major impetus behind the EnergyPlus effort (Crawley et al., 2001).

The component based modeling approach embodied in TRNSYS has been advocated and used in other implementations for over 20 years (Augenbroe, 1986), and it might seem natural that the object-oriented computer programming paradigm, where classes represent components and are connected to each other through well defined interfaces (sometimes called ports), would be well-suited to the component-based modeling paradigm. However it appears that using object-orientation to develop

flexible, multi-domain building simulation tools is difficult and that progress on this front has been fitful (Augenbroe, 2003). Projects such as IDA (Sahlin et al., 2004), SPARK (Sowell et al., 2004), and Modelica (Fritzson, 2004) are aware of the distinctions between modeling, computer programming, and simulation. Although TRNSYS is component based (but not object-oriented), it implements its component models as computer algorithms in the programming language FORTRAN. This conflates the creation of a description of a system – a modeling task – with instructions on how to operate that system – a computer programming task. This is a critical distinction. The developers of SPARK, in setting out to create an efficient solver for differential algebraic equations (DAEs), realized that the instructions and data organization that a computer needs does not in general correspond to how a human modeler thinks and therefore provided a (largely) non-algortihmic component-based *modeling* language, as opposed to a fully algorithmic *programming* language. The human modeler tells the solver what to solve, but not how to solve it. This modeling language implements its component-based philosophy using the object-oriented paradigm applied to model description, incorporating the instantiation of objects (specific components) from classes (generic components) (Sowell et al., 2004 and LBNL and Ayers Sowell Associates, 2003). In order to *simulate* the model, however, a computer must have an algorithm to follow. Happily it turns out that this algorithm creation may be automated by processing the model, and SPARK provides a facility for this, specifically converting the model into C++ source code (LBNL and Ayers Sowell Associates, 2003).

Components in SPARK are defined in "atomic classes" which are implemented as descriptions using C++ syntax. These atomic classes represent the behavior of a component via all possible forms of relevant assignment statements. For example, an atomic class representing resistors may be described by the equation $V = IR$. Even though the resistance R may be known, the computer in general does not know if it needs to assign the product IR to the variable V or the quotient $\frac{V}{R}$ to the variable

I since this depends on the particulars of how this resistor is connected to other components. Therefore, each atomic class must explicitly list all possible incarnations of $V = IR$: $V := IR$ and the inverses $I := \frac{V}{R}$ and $R := \frac{V}{I}$, where the symbol $:=$ indicates an assignment statement in contrast to the symbol $=$ which merely expresses equality (LBNL and Ayers Sowell Associates, 2003). Note the distinction made here between equations and assignments – an equation with n variables is a general expression that may be manifested as n different assignment statements. This feature is critical. This enables acausal modeling by leaving all options open as to what variables need to be solved for, something which is determined when the model is transformed into source code. Although the C++ syntax is used to define classes, it is done in a non-algorithmic, declarative way, in contrast to TRNSYS. This equation-based implementation is a feature common to the modern component-based, object-oriented modeling tools and allows the modeler to focus on describing what the model is rather than how to solve the problem, as well as enabling greater code reuse and easier code maintenance.

The SPARK suite, with its component- and equation-based acausal approach coupled with an efficient solver, would seem to be capable of being the hoped-for flexible and multi-domain tool since it is capable of solving any continuous system problem – possibly with discrete events, in a limited fashion – that can be represented by DAEs, which includes buildings and their constituent parts and physical processes. However, SPARK is foremost a general DAE solver which happens to include a modeling language, and this language is relatively verbose and cumbersome. Atomic classes, in C++, can be combined into macro classes and thus enable hierarchical modeling: the organization of basic low-level classes into more complex higher-level classes. For example two atomic classes of a resistor and capacitor could be combined into a lowpass filter macro class. While hierarchical modeling is beneficial, these macro classes are defined in separate files using a different syntax than C++. The model to be solved, consisting of atomic and macro classes and the links between them, are defined in a

problem file using a syntax similar to that used to define marco classes. Furthermore, another input file with its own format must be included to provide the initial inputs. In addition, the equation inverses need to be explicitly written, although this task can be automated using the symbolic processor packaged with SPARK or third-party programs such as *Maple* (LBNL and Ayers Sowell Associates, 2003).

Modelica, on the other hand, is solely a model description language whose philosophy is similar to that of SPARK, but is more expressive and is implemented with a cleaner and more unified syntax. The Modelica specification is independent of solvers, so the creation of tools to solve models described in Modelica are left to developers. It has been developed by a group of workers associated with several other object-oriented modeling languages such as Neutral Modeling Format (NMF), Omola, and Smile, and tools such as IDA in an attempt to learn from these past efforts and create a well-designed language (Fritzson, 2004 and Sahlin et al., 2004). Today Modelica has reached a level of maturation and stability sufficient to be employed in a variety of industries and could become a *de facto* standard modeling language.

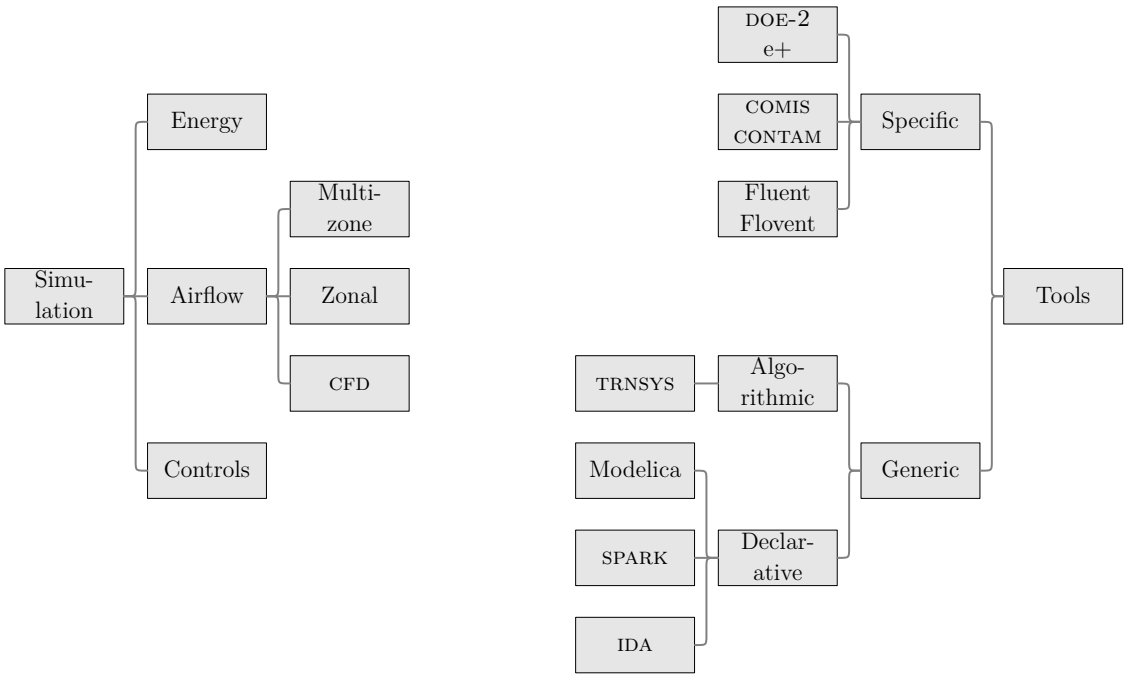


Figure 1.2.1 The Simulation Landscape with All Tool Types

1.3 Modelica, Briefly

Modelica describes physical models in a component-based way, using object-orientation as an organizing method to describe what a model is rather than as a paradigm for specifying how to solve a model. A component is an instance of a class, which represents a generic description of a given model. A class contains both data and expressions describing how that data is related. Being an equation-based language, these expressions are typically declarative equations rather than assignments or sequences of assignments – i.e. algorithms – on that data, which permit acausal modeling and rendering the specification of data flow directions and sequences unnecessary. Being object-oriented, inheritance is supported, so that a subclass can be defined from a superclass and inherit the data and relations of that superclass. The relations between components in particular and model classes in general are described by a special type of class, the connector class which define interfaces, a.k.a. ports or connectors. Classes can be grouped together into packages, forming libraries which along with object-orientation and equation-based, declarative, acausal modeling facilitates the distribution, use, re-use, and maintenance of those classes. The public-domain Modelica Standard Library includes hundreds of models for use in modeling electrical, mechanical, thermal, etc. systems.

Because model behavior is described using mathematical equations, any phenomena which can be described mathematically can in principle be described with Modelica. The language features sufficient expressiveness that continuous, discrete-event, and hybrid (containing both continuous and discrete-event) processes can be modeled. All of these features together mean that models incorporating multiple domains can be constructed. For example a voltage source, resistor, inductor, electrical ground, and electromotive force can be combined to form a motor model. This motor can be a submodel of a vehicle model which connects the motor via a shaft connector to a wheel submodel which propels the vehicle over a rough road. An active suspension

model consisting of a control subsystem, springs, and active dampers can be included in this vehicle model to control ride quality. Once this larger model – combining electrical, mechanical, and control components – has been simulated by a Modelica compiler, one can inspect variables such as motor current and torque, spring response, and vehicle speed.

Although equations are the primary means of describing behavior, Modelica supports the use of algorithms within the language for those times when an assignment or sequence of assignments are more appropriate. Functions may be declared, making general purpose relations such as trigonometric functions available to be called from within classes and models. In addition, external functions written in FORTRAN or C may be called (Fritzson, 2004).

Almost everything in Modelica is a class, and in order to give the language a semantic richness, there are different kinds of classes used for different purposes. The most basic kind of class is a `class` which contains data and equations (and possibly algorithms). There are other kinds of restricted classes as well:

1. `model` – general purpose, similar to `class` but cannot be used in connections
2. `block` – has fixed causality, i.e. member variables are identified as `input` or `output`; cannot be used in connections
3. `record` – has no equations/algroithms and is similar to a `struct` in C; cannot be used in connections
4. `type` – used to define user-defined types
5. `connector` – has no equations/algroithms and is used to define connections between objects
6. `function` – similar to a function in C or Matlab; quite restricted compared to a basic `class`
7. `package` – manages namespaces and creates libraries

Predefined types are *Real*, *Integer*, *Boolean*, and *String*. Arrays may constructed of these basic types or out of user-defined types.

To illustrate some of these features, consider the case of a metal cylinder being heated through its base. We wish to control the heater so that the cylinder remains between 315K and 317K (107°F/41.9°C and 112°F/43.9°C). We will build a Modelica model of this system and illustrate the key points of the preceding description of Modelica. This example could be created entirely with existing models in the Modelica Standard Library, but for present purposes the models will be made from scratch. This simple example's sole purpose is to illustrate Modelica, not build general or highly reusable models.

First use is made of the SI unit definitions in the Modelica Standard Library and a heat-transfer interface between objects is defined:

```
import SIu = Modelica.SIunits;
connector HeatConnector
 SIu.Temperature T     "Temperature";
 SIu.Area A          "Area through which heat moves";
 flow SIu.HeatFlowRate Q  "Heat flow rate";
end HeatConnector;
```

The line which imports the SI unit definitions makes available specialized types that contain information on units; for example they type `SIu.Temperature` is a *Real* that is forbidden from being negative and is identified, through the use of a string annotation, as having the unit of Kelvin. An instance of `HeatConnector` may be created with the line `HeatConnector Foo`, and the member variables accessed using the dot notation, e.g. `Foo.A`. The `flow` qualifier indicates to a Modelica compliler that at each node which is an instance of `HeatConnector`, all occurances the variable `Q` will sum to zero in the manner of Krichhoff's current law. The equations required for this are generated by the Modelica complier.

The strings between the double quotes are optional comments for the benefit of the user of this `connector` and may be used by Modelica implementations in user interfaces. Modelica also supports C style comments for the benefit of the programmer which do not appear in user interfaces.

It is noted in passing that modeling convention encourages that connectors should only contain variables that `flow` or create the potential for `flow`. The presence of area in this connector is applied here as flows will be associated with different areas in the models to follow; convention is thus violated, and furthermore this area can in principle change during a simulation, although areas will be defined such that this will not be the case.

The cylinder will be created from scratch in a simple way based on the First Law of Thermodynamics; it could be modeled with more detail using 'off the shelf' models involving conduction as well as convection and thermal storage in the Modelica Standard Library. We take the cylinder to be simply a mass with density ρ, volume V, and a specific heat (at constant pressure) c_p and temperature T. The First Law of Thermodynamics states that the sum of the energy entering a system minus the sum of the energy going out equals the change of energy in the system. We adopt the sign convention used in Modelica that a quantity flowing into a system is taken to be positive. In rate form, this is:

$$\sum \Delta \dot{E}_{sys} = \sum \dot{E}_{in} + \sum \dot{E}_{out} \tag{1.1}$$

Here, the system is the cylinder. Thus we know $\sum \Delta \dot{E}_{sys}$ and can write

$$\rho V c_p \frac{dT}{dt} = \sum \dot{E}_{in} + \sum \dot{E}_{out} \tag{1.2}$$

To make the Modelica model, first we make an assumption about how many 'pathways' there are for power to flow in or out: two. This is restrictive, and is only adopted for current illustrative purposes.

```
model ThermalCylinder
 HeatConnector BasePort;
 HeatConnector TopNSidesPort;
 SIu.Area A             "Surface area of the cup";
 SIu.Temperature T          "Temperature";
 SIu.Height h            "Height of cylinder";
 parameter Real r2h = 0.75     "Ratio of cylinder radius to cylinder height";
 parameter SIu.Volume V      "Volume";
 parameter SIu.Density rho = 7272     "Density";
 parameter SIu.SpecificHeatCapacity cp = 420 "Specific heat, constant pressure";
 constant Real PI = 3.142;
equation
 /* assume that the Biot number is « 1,
 /* therefore temperature is uniform throughout the cylinder */
 BasePort.T = TopNSidesPort.T;
 T = BasePort.T;
 // some geometry
 V = PI*h*(r2h*h)^2;
 A = 2*PI*(r2h*h)^2 + 2*PI*(r2h*h)*h;
 // assign the appropriate areas to the ports
 BasePort.A = PI*(r2h*h)^2;
 TopNSidesPort.A = A - BasePort.A;
 // first law of thermodynamics
 rho*V*cp*der(T) = BasePort.Q + TopNSidesPort.Q;
end ThermalCylinder;
```

Several of the variables are qualified by `parameter`, indicating that they may change between simulations but not during a simulation. A similar qualifier `constant` can be used for true constants, such as the gravitational constant of the universe, or in this case, $\pi =$ `PI`. Most of the parameters have default values given for them, in this case for cast iron, but these can be changed elsewhere. The volume `V` is not explicitly given a value, but will default to zero. A more appropriate value will be

given later. The other variables are truly variable and can change during a simulation as appropriate.

Note that in the equation `V = PI*h*(r2h*h)^2` and given that the cylinder volume `V` will be specified, it is the cylinder height `h` that is unknown. Because this is an equation and not an assignment statement, Modelica will figure out that `h` needs to be solved for and will not see this line as a command to put the product `PI*h*(r2h*h)^2` into `V`.

The core behavior of this model is expressed in the equation `rho*V*cp*der(T) = BasePort.Q + TopNSidesPort.Q`, the First Law as applied to this model. The time derivative of temperature dT/dt is expressed as `der(T)`.

As an initial test of this model, consider the case of a warm cylinder suspended in cool air, thus simply cooling by convection. In order to model convection using Newton's law of cooling $Q = hA(T - T_\infty)$, the following model may be used...

```
model Convection
 HeatConnector Port;
 parameter SIu.Temperature AmbientT = 295      "Ambient temperature";
 parameter SIu.CoefficientOfHeatTransfer h = 30 "Convection coefficient";
equation
 Port.Q = h*Port.A*(Port.T - AmbientT);
end Convection;
```

...and connected to the cylinder as follows:

```
model CoolingCylinder
 ThermalCylinder  CastIronCylinder(V=0.0003, T(start=316));
 Convection     ConvAtBase;
 Convection     ConvAtTopNSides;
equation
 connect(CastIronCylinder.BasePort, ConvAtBase.Port);
```

```
 connect(CastIronCylinder.TopNSidesPort, ConvAtTopNSides.Port);
end CoolingCylinder;
```

Here a cylinder is instantiated with a volume of 0.0003 m^3 and with an initial temperature of 316K. Convection models intended for the base of the cylinder and the sides and top are instantiated, and are connected to the cylinder in the `equation` section using the `connect` function which generates the appropriate equations for the ports: sum to zero equations at the nodes for the `flow` variables and equations expressing the equality of the non-`flow` connection variables.

This situation is valid, and has an exact solution, provided that the Biot number $Bi = \frac{hL}{k} \ll 1$. For this case, with $h = 30\ \frac{W}{m^2K}$, using the ratio of volume to surface area as the characteristic length $L = 0.012\ m$, and the thermal conductivity $k = 52\ \frac{W}{mK}$, then $Bi = 0.0068$; thus conduction within the cylinder is sufficiently high relative to convection off the surface of the cylinder that surface temperature and interior temperature may be taken to be equal and therefore using `ThermalCylinder` is appropriate. The exact solution is then (Lienhard(IV) and Lienhard(V), 2005)

$$T = e^{-t/\tau}(T_i - T_\infty) + T_\infty \tag{1.3}$$

where t is time, τ is the time constant $\tau = \frac{\rho c_p V}{hA}$, T_i is the initial temperature, and T_∞ is the ambient air temperature.

The `CoolingCylinder` model was simulated using OpenModelica v.1.4.4 (PELAB, 2008) which converts the Modelica models down to C++ source code which is itself built and executed. The time evolution of the temperature of the cylinder is given at selected points in the following table:

Table 1.3.1 CastIronCylinder.T – Cylinder Temperature

time, s	Modelica, K	Exact solution, K
0	316.00000	316.00000
120	314.01405	314.01406
360	310.58783	310.58784
600	307.77899	307.77900
840	305.47628	305.47630
1080	303.58851	303.58853
1320	302.04091	302.04093
1560	300.77218	300.77219
1800	299.73206	299.73208
2040	298.87937	298.87939
2280	298.18033	298.18034
2520	297.60725	297.60726
2760	297.13744	297.13745
3000	296.75229	296.75229
3240	296.43653	296.43654
3480	296.17768	296.17768

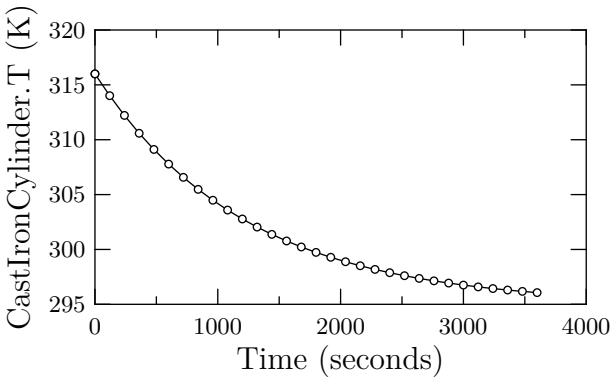


Figure 1.3.1 Temperature of the Cooling Cylinder

This example will now be extended to include an electrical heater with discrete events to form a hybrid model in which the cylinder is maintained between 315K and 317K. This model will be built from generic models using some of Modelica's inheritance features. The models to follow are extended from a simple circuit example

found in the Modelica 3.0 specification (The Modelica Association, 2007) and in Fritzson (2004).

First a few basic classes are defined:

```
connector Pin  "An electrical pin"
 SIu.Voltage v         "Voltage";
 flow SIu.ElectricCurrent i  "Current";
end Pin;
/*==================================*/
class Ground
 Pin p;
equation
 p.v = 0;
end Ground;
/*==================================*/
partial class TwoPin
 Pin   p, n;
 SIu.Voltage deltav   "Voltage between the pins";
 SIu.ElectricCurrent i  "Current";
equation
 deltav = p.v - n.v;
 0 = p.i + n.i;
 i = p.i;
end TwoPin;
```

The Modelica `class` is used here for illustrative purposes, but `model` could just as easily be used. The `TwoPin` class prefixed with the keyword `partial`, which indicates that this class is incomplete: there is no constitutive relation which defines the voltage. This partial class is used to form useful models:

```
class DCVoltage
 extends TwoPin;
 parameter SIu.Voltage va = 50 "DC voltage";
```

```
equation
 deltav = va;
end DCVoltage;
/*==================================*/
class ThermalResistor
 extends TwoPin;
 HeatConnector HeatPort;
 parameter SIu.Resistance R = 50 "Electrical resistance";
equation
 deltav = R*i;
 HeatPort.Q = -R*i^2; // outgoing power (as heat) < 0
end ThermalResistor;
```

The `extends` prefix indicates that both `DCVoltage` and `ThermalResistor` inherit `TwoPin`'s member variables and equations and extends them with new relations. Some connections are defined, specifying them as either `input` or `output`:

```
connector TportIn = input Real;
connector TportOut = output Real;
```

One of these connectors is used to extend the `ThermalCylinder` model:

```
model CntrlThermCyl
 extends ThermalCylinder;
 TportOut Tcon      "Temperature";
equation
 Tcon = T;
end CntrlThermCyl;
```

And now the temperature of the cylinder can be sensed in order to command a switch for the heater:

```
class ControlSwitch
 extends  TwoPin;
```

```
 TportIn  Tcon          "Temperature";
 Boolean  On(start=false);
 Boolean  Off(start=true);
 parameter SIu.Temperature LowTthresh = 315 "Lower threshold temperature";
 parameter SIu.Temperature UpTthresh = 317  "Upper threshold temperature";
 parameter SIu.Resistance Ron = 0.00001   "Electrical resistance when on";
 parameter SIu.Resistance Roff = 100000   "Electrical resistance when off";
equation
 On = Tcon <= LowTthresh;
 Off = Tcon > UpTthresh;
 deltav=i*(if On then Ron else if Off then Roff else Roff);
end ControlSwitch;
```

To avoid numerical difficulties, this switch is modeled as a discretely variable resistor which follows the practice used in a similar model from the Modelica Standard Library where the on condition is modeled by a low resistance and the off condition by a high resistance. The *Boolean* variable *On* changes its states at discrete events, and the two different operating regimes of the switch are encoded in the `if` expression. This introduces events – which are considered to take no 'wall clock' time – into the following, otherwise continuous-time model:

```
model HeatedCylinder
 DCVoltage    DC;
 ControlSwitch  Switch;
 ThermalResistor Heater;
 Ground     GND;
 CntrlThermCyl  CastIronCylinder(V=0.0003, T(start=316));
 Convection   ConvAtTopNSides;
equation
 connect(DC.p, Switch.p);
 connect(Switch.n, Heater.p);
 connect(Heater.n, GND.p);
 connect(GND.p, DC.n);
```

```
  // the area of the heater = area of the base of the cylinder:
  connect(Heater.HeatPort, CastIronCylinder.BasePort);
  connect(CastIronCylinder.TopNSidesPort, ConvAtTopNSides.Port);
  connect(CastIronCylinder.Tcon, Switch.Tcon);
end HeatedCylinder;
```

Textually determining the topology between components (submodels) of larger models can be difficult, and for this reason most Modelica implementations use a graphical representation for the depiction and creation of model topology:

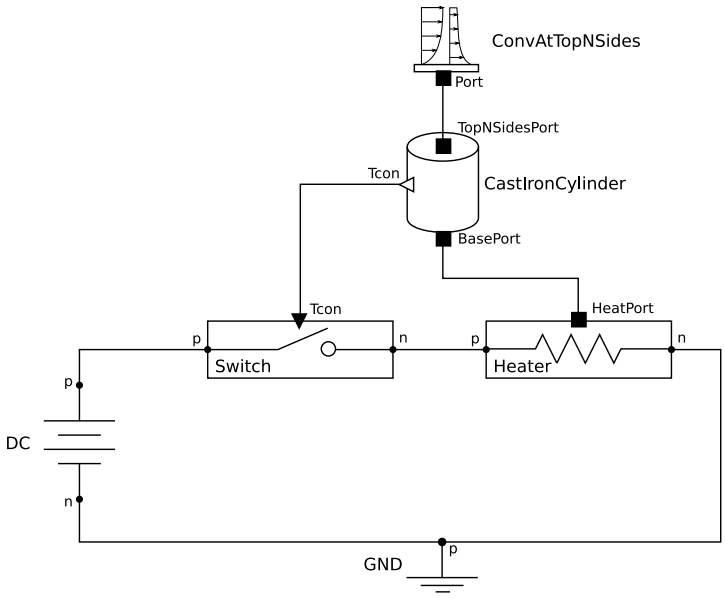


Figure 1.3.2 The HeatedCylinder Model

One of the benefits of Modelica is that model topologies easily match that of the physical artifact.

When this is simulated for 200 seconds in OpenModelica v.1.4.4, we see that the cylinder temperature is maintained between 315K and 317K:

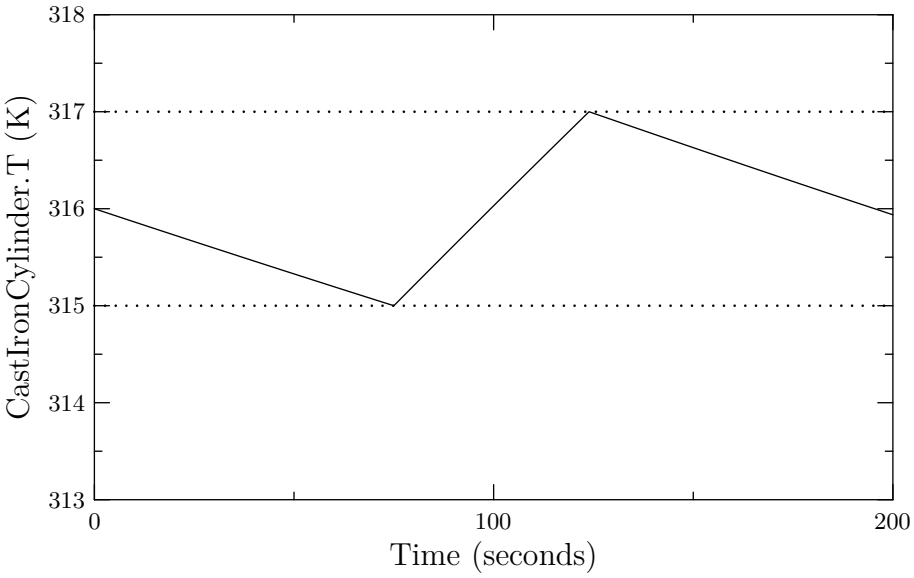


Figure 1.3.3 Controlled HeatedCylinder Temperature

Much of the depth of Modelica was left out of this example: it does not display acausality and leaves this aspect of Modelica unexplored; algorithms and functions, both internal and external, were not treated; the inheritance features were only skimmed despite being a rich and central part of the language; the use of packages and subpackages to create reusable libraries was ignored; the use of annotations, units, and quantities to store model metadata and enrich the functionality of a Modelica tool was unmentioned; and the capabilities to handle events was briefly introduced without substantive explanation. These omissions, made in the name of brevity, conspire with the *ad hoc* nature of the (sub)models to prevent their having high reusability.

However, this simple example demonstrates the breadth of Modelica: object-orientation, equation-based modeling, the integration of multiple domains – electrical and thermal in this case, and the mixing of continuous dynamics with events. Object-orientation allowed the creation of many models simply by extending more general, pre-existing models. Being equation-based, Modelica allows the modeler to focus only on the model rather than on both the model and the steps needed to solve the model. The component-based outlook with well-defined interfaces between them is concor-

dant with how humans think of systems, allowing the rapid – and relatively easy – creation of models from smaller parts as well as the linking of models from different domains. The ability to handle events brings domains with discontinuous dynamics, such as controls, within Modelica's compass.

Modelica is most often used for lumped-parameter modeling, in which the nature of any spatial variations are assumed and only time derivatives explicitly appear. The example given above is a lumped parameter model: spatial variations are assumed to be nonexistent which is reasonable only for a Biot number much less than one. For situations in which there is a spatial variation, lumped parameter models may still be used by assuming a specific form of the spatial variation. For example, if the cylinder was replaced by a flat plate, infinite in extent (the y and z directions) but finite in thickness (the x direction), a model could be defined which implicitly assumes a linear temperature profile across the plate by taking the heat flux $q_x = -kdT/dx = -k\Delta T/\Delta x$, ΔT being the temperature difference between the faces of the plate. In fact this is what the one-dimensional `ThermalConductor` model in the Modelica Standard Library does. However, in unsteady-state conditions the temperature profile may not be linear, yet this can still be approximated in lumped parameter modeling if the family or package of models is well thought out: the `ThermalConductor` model does not include storage effects and must be coupled with a `HeatCapacitor` model, also from the Modelica Standard Library. Separating these two phenomena into different models allows a modeler to link many such models together to approximate the actual, possibly nonlinear, temperature distribution. Another alternative is to create models using integral analysis as described by Batteh (2006). More fundamentally, work has recently been done to enable the solution of partial differential equations in Modelica at the language level, allowing problems with spatial derivatives to be handled in ways similar to what is done with time derivatives (Saldamli et al., 2005).

Wetter and Haugstetter (2006) have compared the development time of similar building energy models expressed in C++ and Modelica, and examined the simulation of similar models in TRNSYS and Dymola (Brück et al., 2002), a commercial Modelica implementation. The time to develop models was decreased by up to an order of magnitude by using Modelica rather than C++; simulation time in Dymola was up to four times slower than in TRNSYS, although it was judged, based on similar studies using SPARK and IDA that this simulation time should not be viewed as an inherent feature of equation-based modeling. As a further example, Wetter (2006a) was able to quickly create a Modelica multizone airflow library which compared well to CONTAM.

1.4 Claims and Goals

With the aim of integrating the disparate building simulation subcultures, this work proposes to build a CFD package for Modelica to bridge CFD airflow simulations with energy simulation and controls.

It is claimed that equation-based acausal modeling can be used to create models which when connected will couple CFD and heat transfer into a (quasi) conjugate model of conductive and convective heat flow which can also interact with other domains, for example controls. These models can co-exist at different levels of spatial resolution, for example lumped conduction in the wall can be coupled to high-resolution CFD models. Furthermore this coupling can be achieved without the need to explicitly code iterative procedures to ensure consistency between conductive heat transfer and CFD solutions, or without the need for one model, CFD for example, to be concerned with the representation of the coupled physics in another model, for example how conduction is modeled in a wall. In addition a particular method of CFD based on the Boltzmann equation is a suitable technique which facilitates this coupling.

One point should perhaps be addressed. Djunaedy et al. (2005) has argued that linking energy simulation capabilities with airflow simulation capabilities should be done by interfacing pre-existing programs together, a.k.a. external coupling, since this avoids the rewrite of code and allows each individual program to advance at its own pace and under its own mechanisms. There is some merit to these points, and it may appear that what is proposed here goes against them. It does in the first case, it does not in the second. First, although this work would be a from-scratch capability, the effort is worthwhile because of the modeling possibilities and modeler benefits that Modelica can provide. Rewriting – or expressing anew – a model in Modelica is fundamentally different than rewriting a CFD algorithm within the EnergyPlus simulation kernel. Second, it is important to bear in mind that Modelica is not a solution, but a platform for solutions. Packages of Modelica models can evolve in their own domains so long as the interfaces between models remain synchronized, which is also necessary for integrating separate programs written in programming languages. That such packages can evolve independently forms a secondary hypothesis.

The goals of this work are to provide an initial capability at linking CFD to the different domains in building performance simulation – fluid flow, heat transfer (energy), controls, etc. within one flexible modeling and simulation paradigm and identify avenues of future research.

1.5 Scope of Work and Methodology

This work will use largely existing CFD techniques to develop an initial capability completely within the Modelica language. Comparisons will be made to flow situations for which there are analytical solutions or benchmark data in the literature. Only laminar flows are considered despite the importance of turbulence; however turbulence is discussed at points in this thesis.

1.6 Thesis Structure

- Chapter 1 provides background on and motivation for multi-domain simulation and introduces equation-based modeling. The Modelica language is also introduced.

- Chapter 2 reviews work on coupling CFD to energy simulation ES and to other modes of heat transfer. The basics of the lattice Boltzmann technique are described, along with various lattice Boltzmann models, boundary conditions, and previous couplings of lattice Boltzmann simulations to non-convective heat transfer simulations.

- Chapter 3 gives an overview of the CFD model as implemented in Modelica.

- Chapter 4 reports on the results of simulations using this model, both for purely fluid-mechanical problems and for problems coupling CFD to conduction heat transfer.

- Chapter 5 discusses these results and the contribution they represent, as well as illuminating the limitations of this work and pointing out directions for future research.

- Chapter 6 summarizes the work and makes specific recommendations.

CHAPTER 2: COUPLING CFD TO NON-CONVECTIVE HEAT TRANSFER PROCESSES

The macroscopic equations expressing the conservation of mass, momentum, and energy in terms of the velocity vector $\boldsymbol{u}$, pressure p and temperature T in a fluid have no general solution and only a few analytic ones. Analytic solutions for some flow situations can be found if the problem permits simplifications of the governing equations. For generality and resolution of the field variables, numerical solutions of the field equations are sought. In the case of a time-varying non-isothermal flow of an incompressible Newtonian ideal fluid where the Boussinesq approximation applies, these equations are respectively:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.1}$$

$$\rho\left[\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}\right] = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{F} \tag{2.2}$$

$$\rho c_p\left[\frac{\partial T}{\partial t} + (\boldsymbol{u} \cdot \nabla)T\right] = \frac{\partial p}{\partial t} + (\boldsymbol{u} \cdot \nabla)p + \nabla \cdot (k\nabla T) + \Phi \tag{2.3}$$

Here $\boldsymbol{F}$ is a buoyancy force term Φ is a dissipation function. The first equation, expressing the conservation of mass, is also called the continuity equation; the set of equations represented by the second vector expression are collectively referred to as the Navier-Stokes equations; the final equations is the conservation of thermal energy, which in fluid mechanics is referred to simply as the energy equation (Currie, 1993 and Tritton, 1997). Several techniques exist for numerically solving these equations (Tannehill et al., 1997), but regardless of the technique, this general approach is a 'top-down' one: the governing equations as expressed at the macroscopic level are solved.

There is a framework for solving partial differential equations in Modelica (Saldamli et al., 2005) that in principle could be used to solve the equations above.

However, this framework currently only has support for finite differences and finite elements, whereas the preferred paradigm for 'top-down' CFD is finite volumes. Furthermore this framework relies on code external to Modelica.

There is, however, a 'bottom-up' CFD approach (Kadanoff, 1986) which postulates a model of a microworld whose rules, during the course of their operation, mimic the interactions of fluid molecules and *recreates* at the macro level the behavior expressed by the continuity, Navier-Stokes, and energy equations. This approach is derived from the Boltzmann equation describing microscopic particle interactions which like the macroscopic equations above is difficult to solve analytically. The approach does not solve the exact microscopic dynamics as this would require unimaginable computing resources (Cercignani, 1988), but only simulates a reduced version of these dynamics sufficient to be macroscopically accurate, typically over a regular lattice. This approach is therefore called the lattice Boltzmann method (Wolf-Gladrow, 2000, Chen and Doolen, 1998 and Yu et al., 2003).

It is here proposed to use the lattice Boltzmann method to implement the CFD capability for the building airflow suite in Modelica as it is simple and is easy to program, this being a proof of concept study, and because it is naturally suited for unsteady flows which fits well with Modelica's emphasis on time domain analysis of systems. The lattice Boltzmann method does have several advantages: inherent suitability to parallel computing (Chen and Doolen, 1998); specific implementations of this method can have the heat flux at a boundary be an inherent part of the solution, opening up options for the coupling of CFD simulations to heat transfer models such as conduction through walls; and can be used for situations involving mixing (Yu, 2004) as well as multi-component fluids and flows with particulate suspensions (for example see Chen and Doolen (1998) and Yu et al. (2003) and the references therein). A disadvantage is a high memory demand, particularly for three-dimensional flows.

The lattice Boltzmann method has been applied to the building simulation realm

before: Crouse et al. (2002) and Kuhner et al. (2004) simulated the turbulent, thermal flow around and through a house, as well as within an office. Further work incorporated this lattice Boltzmann based method into a collaborative design environment for HVAC system layout (Borrmann et al., 2006). These authors have one similar goal as the present work, coupling CFD to other domains, although they focus on the incorporation of CFD into workflows and on integration with building product models – specifically, the Industry Foundation Classes – and do not use Modelica. Kuznik and Rusaouen (2007) applied a thermal implementation to situations modeling double skin facades and spaces with localized heating elements. On a larger scale, de la Fuente et al. (2003) used the lattice Boltzmann method to investigate simple problems representative of urban airflows.

2.1 Previous Work in Connecting CFD to Other Domains

2.1.1 *Conjugate Approaches: CFD to Non-convective Heat Transfer*

The most direct coupling between CFD and the conductive heat transfer in a wall is to form a conjugate solution in which the equations of fluid motion and conductive (and/or radiative) heat transfer are solved simultaneously in one greater model. Such an approach involves solving these equations at similar or the same level of spatial resolution and amounts to what is in effect – if not necessarily in actual execution – the combination of a CFD model with, e.g., a finite-element model of the heat diffusion equation with identical dimensionality and similar grid spacing.

Conjugate methods have applications in many fields, and have been applied to building spaces. Chen et al. (1995) conducted a 3D unsteady conjugate study on a room with one window and one radiator, solving the equations of fluid flow (continuity, Navier-Stokes, and energy), conduction through the walls, and radiation exchange

between the walls with results showing good comparison with experimental data. Similarly Potter and Underwood (2004) developed a conjugate modeling method for a room and simulated cases of forced and natural convection with conduction and radiation heat transfer. Ben-Nakhi and Mahmoud (2007) used a conjugate method to simulate the flow and heat transfer processes in a realistic model of a roof cavity in summer, following up this work with a similar case for winter (Ben-Nakhi and Mahmou, 2008).

All of the previously mentioned studies solved the fluid flow and conduction equations at the same dimensionality and similar spatial resolution levels. Kaminski and Prakash (1986), in contrast, investigated conjugate heat transfer in which different levels of dimensionality and spatial resolution were used in the models of conduction in a vertical wall of a square two dimensional cavity. Results show that in many cases solving the conduction problem at a lower dimensionality and lower resolution levels can lead to results with only a small error compared to higher dimensionality and resolution simulations of wall conduction.

2.1.2 *Non-Conjugate Approaches:* CFD *to Building Performance Simulation, via Heat Transfer*

The conjugate approaches mentioned above all solve the fluid flow and heat transfer aspects of a problem at the same dimensionality and the same or similar levels of spatial (and temporal) resolution. While there are situations when such approaches may be appropriate, there are also cases when this is unnecessary or unwanted. Whole-building simulation programs deal with entire buildings, and simulating the heat and fluid flow processes within them at the fine resolution levels alluded to in the conjugate approaches above is computationally expensive and most often unnecessary (see, e.g. Kaminski and Prakash (1986) as previously mentioned for simulation results which lead to this conclusion). However there are cases when a combined simulation may

benefit from having two different levels of resolution, such as natural/hybrid ventilation situations or questions of thermal comfort where details provided by CFD simulations are relevant, but the associated heat transfer through walls need not be treated so finely except insofar as the wall conduction affects the flow. Similarly, uncertainties associated with the convective heat transfer through a wall may be reduced via CFD simulations, but the actual heat transfer through the wall itself can for practical purposes be modeled as a lumped one dimensional conduction element. Furthermore, even in a whole building simulation which, for example, natural or hybrid ventilation is present, it is unlikely that CFD is required in every space: perhaps only one space, say an atrium, would need to be modeled using CFD, while the rest are adequately modeled using lower resolution approaches.

Most of the effort at combining CFD simulations to the lower-spatial-resolution heat transfer models such as those used in building simulation tools have been toward conflating two *separate programs* together. Negrao (1995) developed a CFD module for use in the ESP-r building simulation program in what is essentially the internal coupling paradigm, in which a CFD module is developed to be used by a particular building simulation program, forming more or less a unified greater package. In contrast, Zhai (2003) developed a CFD program specifically for incorporation with EnergyPlus, although in a manner that it could be used independently and could evolve independently with a minimum of overhead in the infrastructure to couple to EnergyPlus. Similarly, Djunaedy (2005) and Mirsadeghi et al. (2009) coupled pre-existing and independently developed (sometimes commercial) CFD and building simulation tools together, for example ESP-r and FLUENT. This paradigm is called external coupling.

In all of these works, whether the conflation is internal or external, the separate programs overlap in the building air spaces: the CFD program handles thermal air flows and hence convective heat transfer through nonisothermal air spaces, while

the building (energy) simulation program handles not just conductive heat transfer through walls, but also convective transfer through the air spaces using Newton's law of convective heat transfer assuming mixed, spatially isothermal conditions. Thus, these approaches are not conjugate in the sense described in the previous section, in which there is no overlap. For these overlapping simulation configurations in which the separate programs each model convective heat transfer in air spaces, the challenge is in coupling the two programs together in a physically consistent fashion.

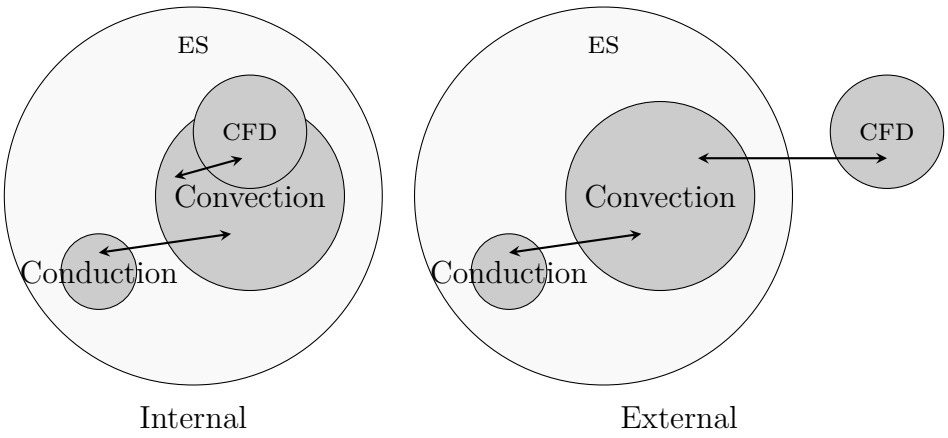


Figure 2.1.1 Existing Coupling Paradigms

It is useful for the sake of context to repeat here the coupling methods and coupling strategies as reported by Zhai (2003). Here, *coupling methods* refers to which variables are exchanged between CFD and the whole building energy simulation (here referred to as ES, in the manner of Zhai). *Coupling strategies* refers to the manner in which CFD and ES exchange these variables as a simulation progresses in time. Note that this terminology is separate from the notion of coupling *paradigm*, which here refers internal or external coupling.

For the coupling methods, the choices of the variables is explained with the help of figure **2.1.2**; h is the convective heat transfer coefficient, found by CFD, between the wall temperature T_{wall} and a temperature close to the wall T_{wall+} also found by CFD; h_{nom} is a nominal convective heat transfer coefficient, also found by CFD, defined

by the temperature difference between T_{wall} and the mixed room air temperature as used by ES.

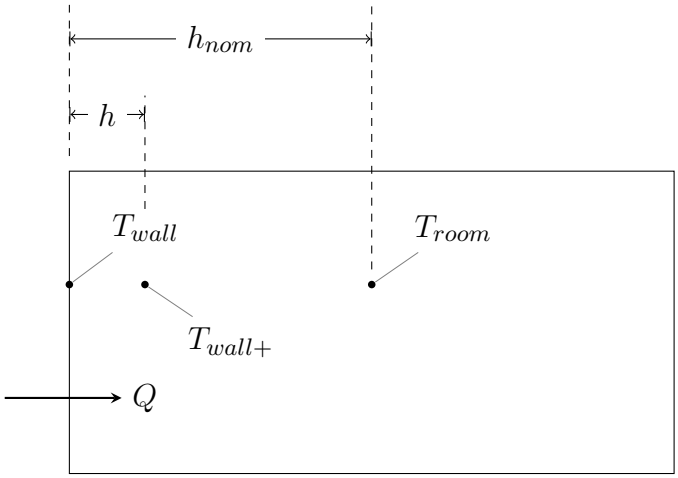


Figure 2.1.2 Variables for ES – CFD Coupling Methods

The objective of using these coupling variables is ultimately to have a consistent solution in temperature and heat flux/heat flow rate between the two overlapping programs. However the methods specific to each program prevent directly coupling temperature and heat, as explained shortly.

The six coupling methods are explained in figure **2.1.3**.

The first 5 methods are studied extensively (Zhai and Chen, 2003, 2004, 2005), however the sixth is dismissed due to the CFD scheme being capable of only Dirichlet (specified temperature) or Neumann (specified heat flux) boundary conditions; a Robin-type boundary condition which relates wall conduction to air convection is absent and so method 6 is effectively out of scope. It is this method that is the focus of the current work, however.

The coupling strategies are grouped into static, dynamic, and bin coupling processes. Static processes exchange variables between ES and CFD only for certain times when

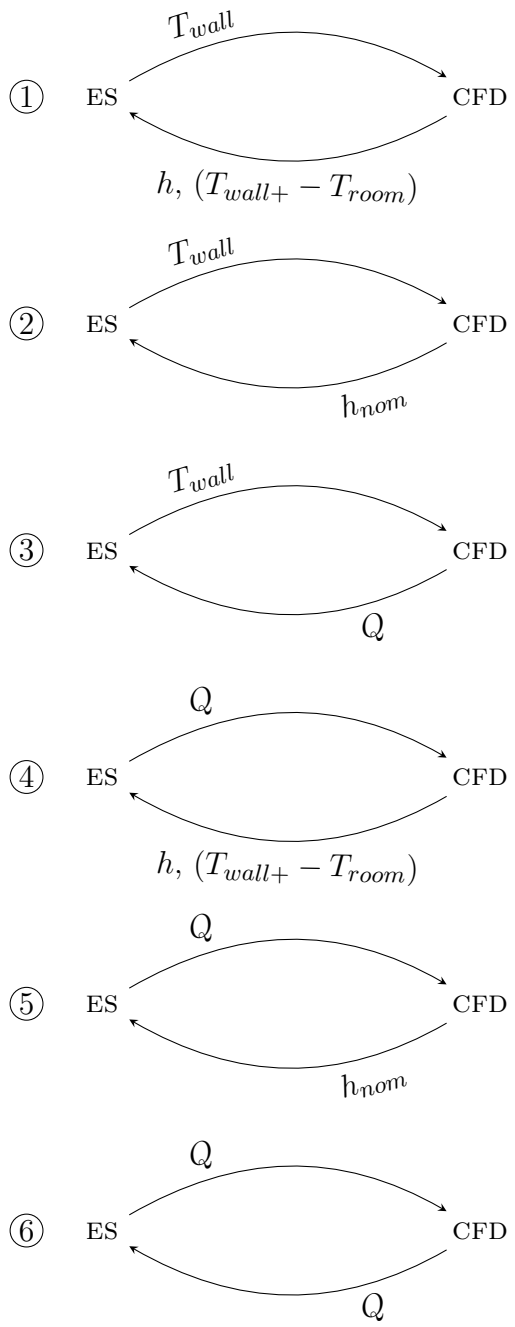


Figure 2.1.3 The Six ES – CFD Coupling Methods of (Zhai, 2003)

judged necessary to improve the solution of ES, CFD, or both; this coupling may be one step (ES → CFD or CFD → ES) or two step (ES → CFD → ES or CFD → ES → CFD). Bin coupling seeks to precompute coupled ES and CFD results for use in an exclusively ES run and can be of a static type (coupled ES and CFD simulations are run for a

variety of situations and the binned results are used in an ES run) or a dynamic type (ES and CFD coupled simulations produce binned results for a limited-time ES run, after which ES and CFD coupled simulations are run anew for a fresh set of binned results for another limited-time ES run).

The dynamic coupling processes are the ones of interest when working with an equation based modeling language such as Modelica. These are depicted in figure **2.1.4**, with the exception of the "one time step dynamic coupling", which is a subset of one of the pictured dynamic coupling processes.

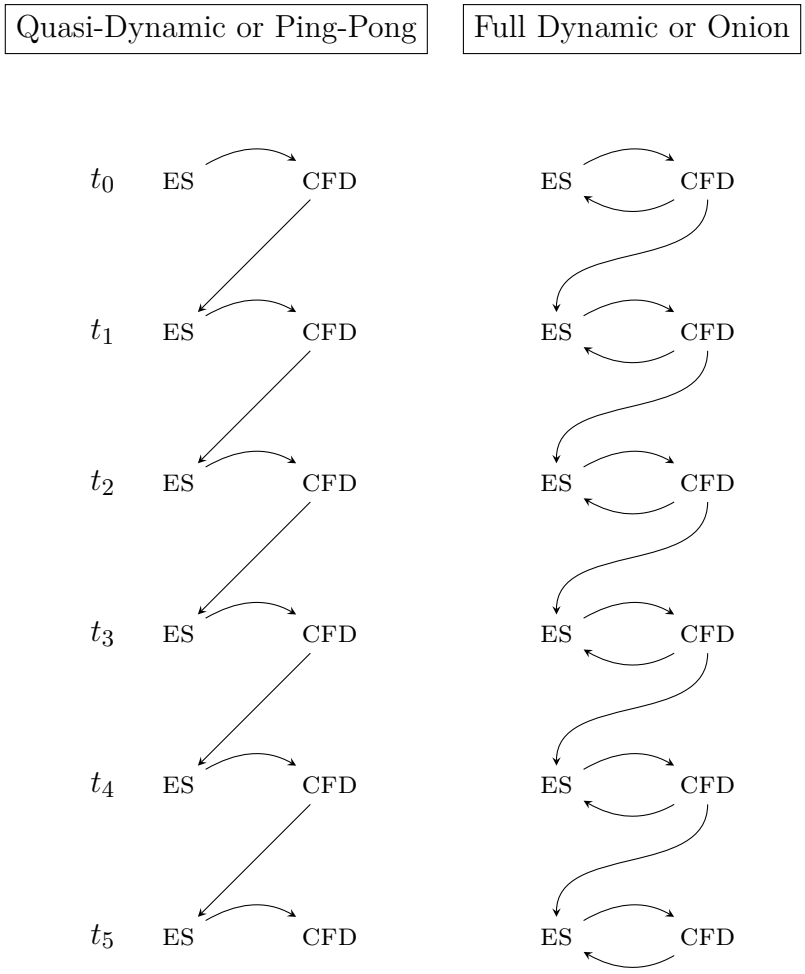


Figure 2.1.4 Dynamic Coupling Strategies

As indicated in the figure, the quasi-dynamic coupling and the full dynamic coupling

are the 'ping-pong' and 'onion' techniques, respectively, used to couple nodal network models to ES (Hensen, 2003); in other work the the ping-pong style approach is referred to as loose coupling (e.g. (Mirsadeghi et al., 2009)). In the quasi-dynamic/ping-pong case, accuracy is a secondary concern and no iteration is done between ES and CFD to achieve a consistent solution in T and Q. However the ping-pong has been shown to result in instabilities when used to couple heat transfer with multizone (nodal) airflow network problems in practical situations (Sahlin, 2003). In the full dynamic/onion case, iteration is performed to ensure that T and Q in ES is the same as T and Q in CFD. In both cases, the results of one program are fed forward to the other program at the next time step.

2.2 Towards An Alternate CFD Approach Based on Kinetic Theory: A Review

It will be argued here that the lattice Boltzmann method has some advantages for use in coupling CFD to building heat transfer, and to make these points some background on this method will be given. Historically the lattice Boltzmann technique is an outgrowth of cellular automata (Frisch et al., 1986 and Wolf-Gladrow, 2000), although the most physically fundamental view is that lattice-Boltzmann is a numerical scheme for the solution of a simplified version of the continuous Boltzmann equation (Sterling and Chen, 1996 and Cao et al., 1997), which describes the evolution of the single particle distribution function f for a (ideal) gas.

Appendix A gives a brief overview of kinetic theory but is summarized here. The Boltzmann equation of interest is:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}_1} = \frac{1}{\theta}(f^{EQ} - f) \tag{2.4}$$

Here $f = f(\boldsymbol{x}, \boldsymbol{\xi}, t)$ is the expected mass density at a point in space $\boldsymbol{x}$ having a molecular velocity $\boldsymbol{\xi}$ at time t; in essence it is the probability of finding a 'particle'

moving with velocity $\boldsymbol{\xi}$ at location $\boldsymbol{x}$ and time t. The term on the right hand side is a simplified model of a complex integral Ω_B which describes the collisions of molecules; θ is the rate at which the system 'relaxes' to local thermodynamic equilibrium, and is the parameter through which material properties such as viscosity are expressed. Equation **2.4** is termed, among other things, the Boltzmann BGK equation, after the originators of the simplified BGK collision term $\frac{1}{\theta}(f^{EQ} - f)$ (Bhatnagar et al., 1954).

The term f^{EQ} is the Maxwell-Boltzmann distribution function

$$f^{EQ} = \frac{\rho}{(2\pi RT)^{(\eta/2)}} e^{\frac{-(\xi-u)^2}{2RT}} \tag{2.5}$$

where η is the degrees of freedom available to the gas molecules given the problem of interest. Given the definition of f, we can see that the macroscopic density of a gas is given by the zeroth moment of f

$$\rho(\boldsymbol{x}, t) = \int f(\boldsymbol{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi} \tag{2.6}$$

and that the momentum is given by the first moment

$$\rho(\boldsymbol{x}, t)\boldsymbol{u}(\boldsymbol{x}, t) = \int \boldsymbol{\xi} f(\boldsymbol{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi} \tag{2.7}$$

Working with some terms in this first moment, the pressure can be shown to be given by the ideal gas equation of state

$$p = \rho RT \tag{2.8}$$

The second moment is

$$\frac{1}{2}\int \xi^2 f d\boldsymbol{\xi} = \frac{1}{2}\rho u^2 + \frac{1}{2}\int c^2 f d\boldsymbol{\xi} \tag{2.9}$$

where $\boldsymbol{c} = \boldsymbol{\xi} - \boldsymbol{u}$ is the peculiar velocity, which is the random molecular motion relative to the bulk macroscopic flow $\boldsymbol{u}$. Intuitively, the second term above would seem to relate to temperature; indeed it can be shown that

$$\frac{1}{2}\int \boldsymbol{c}^2 f d\boldsymbol{\xi} = \frac{1}{2}\rho\eta RT \tag{2.10}$$

The heat flux q in direction $\alpha \in \{x, y\}$ is a part of the third moment; its form in tensor notation is

$$q_\alpha = \frac{1}{2}\int c_\alpha c^2 f d\boldsymbol{\xi} \tag{2.11}$$

There are many other such moments, all of which are physically descriptive if not usually relevant (see, e.g. (Grad, 1949a)). Thus there is a large amount of information just in the single particle distribution function f, if it can be found; particularly attractive is that the heat flux is an inherent part of the solution. The solution to the (continuous) Boltzmann equation is difficult however, in part due to the need to not only solve in time and in *physical* or 'spatial' space, but also in the large and continuous *velocity* space given by all possible values of the molecular velocity $\boldsymbol{\xi}$. The lattice Boltzmann method attempts to tap into the large amount of information in f by solving the Boltzmann BGK equation in discretized temporal, physical, and velocity spaces.

2.2.1 *Introduction to the Lattice Boltzmann Method*

Discrete approaches to solving the Boltzmann equation also go back at least to Broadwell's investigations of shock waves (Broadwell, 1964b) and to Couette and Rayleigh flow (Broadwell, 1964a). Modern approaches view the lattice Boltzmann equation as a specific finite-difference approximation of a Boltzmann BGK equation with discrete velocities $\boldsymbol{\xi}_i$ (Sterling and Chen, 1996 and Cao et al., 1997); or perhaps more completely as the fully-continuous Boltzmann BGK equation integrated in time over the interval δt. Such an integration assuming a small enough δt yields

$$f(\boldsymbol{x}+\boldsymbol{\xi}\delta t,\boldsymbol{\xi},t+\delta t)=f(\boldsymbol{x},\boldsymbol{\xi},t)-\frac{1}{\tau}(f(\boldsymbol{x},\boldsymbol{\xi},t)-f^{EQ}(\boldsymbol{x},\boldsymbol{\xi},t)) \tag{2.12}$$

when terms of $O(\delta t^2)$ are neglected. Here, $\tau=\theta/\delta t$ is the non-dimensional relaxation time.

The discretization of physical space – $\boldsymbol{x}$ – and velocity space – $\boldsymbol{\xi}$ – arises via the requirement that the discretization still leads to the Navier-Stokes equations (only athermal schemes are considered for the moment). Expanding f^{EQ} in a Taylor series in which terms of $O(\boldsymbol{u}^3)$ are neglected yields an approximation of f^{EQ}, namely f^{eq}; an integral involving f^{eq} that is required for this consistency with Navier-Stokes is of a form for which 3^{rd} order Gauss-Hermite quadrature is the ideal solution procedure. For two-dimensional problems, the abscissae of this quadrature define 9 discrete velocities $\boldsymbol{\xi}_i$

$$\begin{aligned}\boldsymbol{\xi}_0&=(0,0)\\ \boldsymbol{\xi}_{1,3}&=(\pm\check{\xi},0)\\ \boldsymbol{\xi}_{2,4}&=(0,\pm\check{\xi})\\ \boldsymbol{\xi}_{5,6,7,8}&=(\pm\check{\xi},\pm\check{\xi})\end{aligned} \tag{2.13}$$

where $\check{\xi}=\sqrt{3RT}$, thus discretizing velocity space. Physical space is discretized congruently with this velocity space discretization, i.e. the nodes of the spatial grid (*where* the solution f is determined) are spaced a distance $\delta x=\check{\xi}\delta t$. For the athermal model considered here, temperature has no meaning and so $\check{\xi}$ is taken to be arbitrary while still satisfying $\check{\xi}=\frac{\delta x}{\delta t}$ for convenience.

This discretization of the velocity and physical spaces also leads to the discretized form of f^{eq}

$$f_i^{eq}(\boldsymbol{x},t)=\rho w_i\left[1+\frac{3}{\check{\xi}^2}\boldsymbol{\xi}_i\cdot\boldsymbol{u}+\frac{9}{2\check{\xi}^4}(\boldsymbol{\xi}_i\cdot\boldsymbol{u})^2-\frac{3}{2\check{\xi}^2}(\boldsymbol{u}\cdot\boldsymbol{u})\right] \tag{2.14}$$

where w_i are the weights of the 3^{rd} order Gauss-Hermite quadrature:

$$w_i = \begin{cases} \frac{4}{9}, & i = 0 \\ \frac{1}{9}, & i = 1,2,3,4 \\ \frac{1}{36}, & i = 5,6,7,8 \end{cases} \tag{2.15}$$

Thus the continuous Boltzmann BGK equation has been converted to

$$f_i(\boldsymbol{x} + \boldsymbol{\xi}_i \delta t, t + \delta t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\Big[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\Big] \tag{2.16}$$

The moments of f for this athermal model are likewise evaluated by quadrature (He and Luo, 1997a, 1997b):

$$\begin{aligned} \rho &= \sum_{i=0}^{8} f_i \\ \rho \boldsymbol{u} &= \sum_{i=1}^{8} \boldsymbol{\xi}_i f_i \end{aligned} \tag{2.17}$$

and the pressure can be determined simply from

$$p = \rho \frac{\check{\xi}}{\sqrt{3}} \tag{2.18}$$

The dimensionless relaxation time τ is related to the kinematic viscosity by

$$\tau = \frac{1}{2}\left(6\nu \frac{\delta t}{\delta x^2} + 1\right) \tag{2.19}$$

the form of this relation is dictated by stability requirements and has the effect of making what is a first order spatial discretization effectively second order (see, e.g. (Sterling and Chen, 1996)).

This is the standard D2Q9 (2-dimension, 9 speeds) model. To summarize, and as shown in figure **2.2.1**, at each node location $\boldsymbol{x}$ on a regular lattice, there are 9 non-equilibrium particle distribution functions $f_i(\boldsymbol{x}, t)$, 9 equilibrium particle distribution functions $f_i^{eq}(\boldsymbol{x}, t)$, and 9 molecular or 'lattice' velocities $\boldsymbol{\xi}_i$, indicated by the subscript $i \in [0, 1, 2, 3, 4, 5, 6, 7, 8]$.

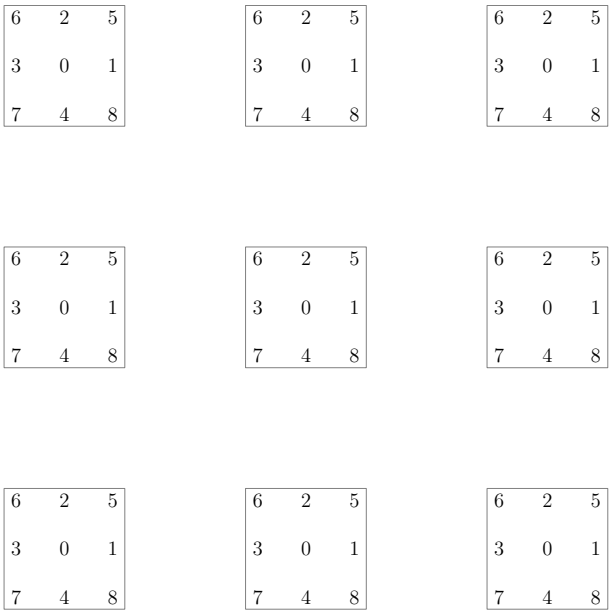


Figure 2.2.1 The D2Q9 Lattice

The operation of the lattice Boltzmann method begins with the first calculation of $f_i^{eq}(\boldsymbol{x}, t)$ and the application of initial conditions which for this first timestep is often, due to convenience despite not being physically correct (Skordos, 1993), taken to also equal $f_i^{eq}(\boldsymbol{x}, t)$. At each timestep the distributions (or 'particles') move along the links of the lattice as defined by the discrete lattice velocities ξ_i as shown in figure **2.2.2**.

After this streaming operation the particles collide according to $\tilde{f}_i(\boldsymbol{x}, t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\Big[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t)\Big]$.

Boundary conditions, in the form of manipulations to the particle distributions which correspond to the macroscopic variables such as $\boldsymbol{u}$, are applied and the distributions are streamed to new locations and the next time according to $f_i(\boldsymbol{x}+\boldsymbol{\xi}_i\delta t, t+\delta t) = \tilde{f}_i(\boldsymbol{x}, t)$. The density and velocity fields are calculated according to equation **2.17**.

At this point the algorithm continues to the next timestep, repeating from calculation of updated equilibrium distributions (without setting $f_\alpha^{eq}(\boldsymbol{x}, t) = f_\alpha(\boldsymbol{x}, t)$), terminating at the user's discretion (Wolf-Gladrow, 2000, Yu et al., 2003 and Yu,

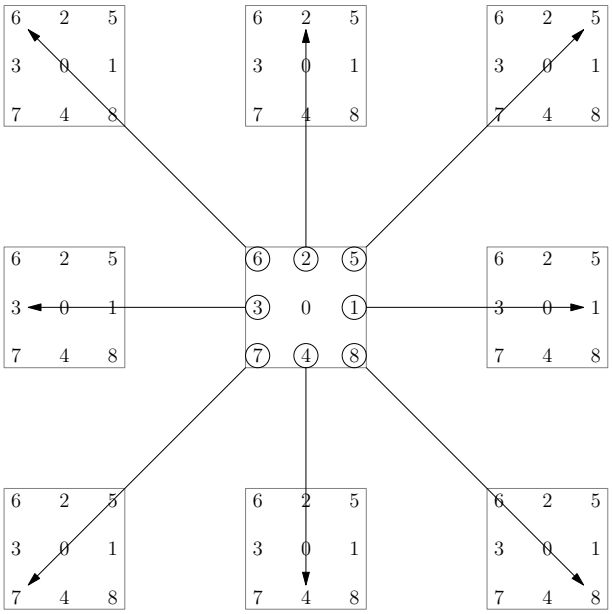


Figure 2.2.2 Streaming Along Links of the D2Q9 Lattice

2004).

In contrast to solving the continuity and Navier-Stokes equations, in this athermal lattice Boltzmann technique there are no nonlinear terms such as $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$, mass is inherently conserved, and pressure is simply determined using a state equation and without recourse to solving a Poisson equation. Despite its simplicity, this method is inherently appropriate for unsteady flows.

It is noted in passing that lattice Boltzmann models can be constructed in which there is more than on relaxation time τ. These models have advantages such as improvement of stability characteristics and the ability to choose transport characteristics – e.g. viscosity ν, conductivity κ, specific heat c_p, etc. – completely independently of one another (D'Humières, 1994 and D'Humières et al., 2002), something which cannot be done in all lattice Boltzmann models. However it is possible to mimic specific fluids such as air with a single relaxation time model, as will be demonstrated in Chapter 4.

2.2.2 *Early Lattice Boltzmann Thermal Models*

The introductory model given in the previous section was the common or 'standard' two dimensional athermal model. Stable thermal models have been more difficult to construct. Early attempts involved treating the temperature as a scalar whose evolution is described by a separate, traditionally-macroscopic-style transport equation that is coupled to the lattice Boltzmann scheme. Eggels and Somers (1995) used such an approach for free convective flows in a square cavity and obtained results that compared well to benchmark data. Similarly, Shan (1997) simulated Rayleigh-Bénard convection in two and three dimensions by treating temperature outside the lattice Boltzmann scheme by considering it to be a passive scalar which evolves according to an advection-diffusion equation.

The hybrid approach is similar in that a separate equation is used to solve for the evolution of the temperature field, however the temperature is not treated as a scalar but is modeled in accordance with an energy equation similar to equation **2.3**, for example by running a finite-difference simulation of the energy equation in conjunction with a lattice Boltzmann model for the momentum (Filippova and Hanel, 2000). Crouse et al. (2002) used this approach in three-dimensional simulations of flow around and through a realistic house model, and an identical approach was used by Mezrhab et al. (2004) to simulate various benchmark problems. The hybrid approach has also been applied with multiple relaxation time lattice Boltzmann techniques (Lallemand and Luo, 2003).

2.2.3 *Multispeed/Expanded Lattice Thermal Models*

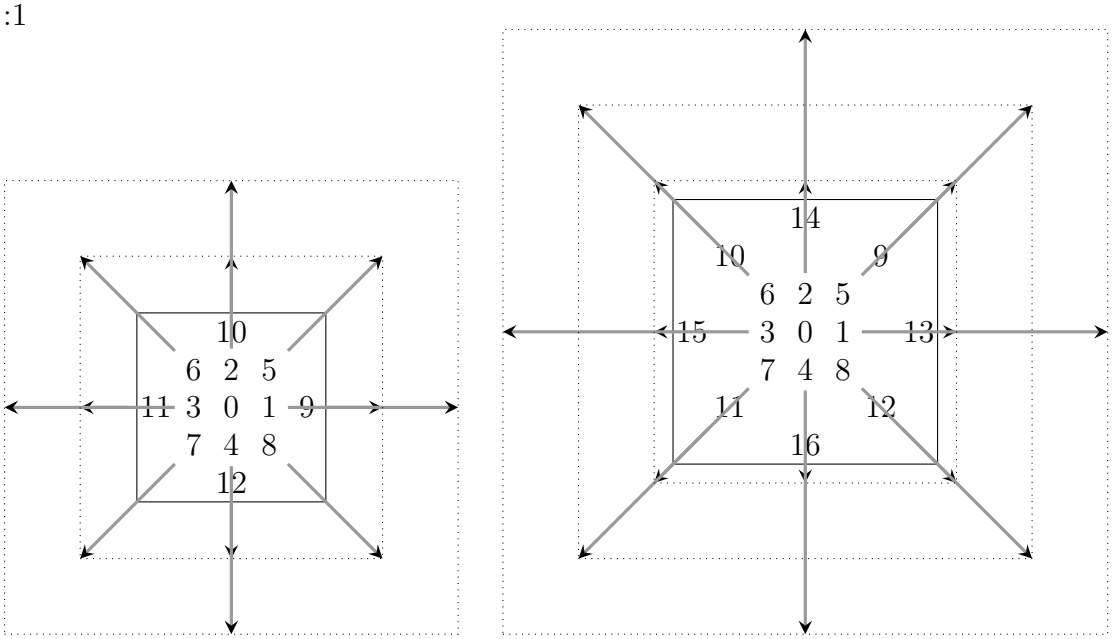
The scalar and hybrid approaches have been successful in accounting for the temperature field, but they do not tap into the information, e.g. of the heat flux in addition to the temperature, that may be had with an approach more closely tied to the

Boltzmann equation. Perhaps the simplest thermal model that is based completely on kinetic theory is the D2Q9 model of Prasianakis et al. (2006). Here the discrete equilibrium distribution f_i^{eq} is developed not from the Taylor-series expansion of the continuous equilibrium distribution function f^{EQ} followed by Gaussian quadrature, but by following a discretized analogue of the origin of f^{EQ} itself: specifically, a discrete form of the H function (see Appendix A) is minimized given the constraints of the conservation of mass, momentum, and energy. Such an entropic model (Karlin et al., 1999 and Ansumali and V. Karlin, 2002) yields a version of f_i^{eq} which incorporates temperature and in fact reduces to the athermal form of f_i^{eq} when the reference temperature of the model is specified. As a result the standard athermal D2Q9 model can be retrofitted to a thermal model merely by swapping the athermal f_i^{eq} for the thermal f_i^{eq}, given the use of common lattice units.

This model, however, contains a fixed Prandtl number of 4, which given air's Prandtl number of 0.71 renders this model inapplicable for present practical purposes. Furthermore, the heat flux is in error, and although both of these problems were remedied in later work by applying correction terms, these corrections involve the calculation of derivatives (Prasianakis and Karlin, 2007).

Use of a "standard lattice" such as the D2Q9 is adequate for the creation of lattice Boltzmann models which accurately describe macroscopic quantities that are second or lower moments of the distribution function f, such as temperature. However the the near-simultaneous work of Philippi et al. (2006) and Shan et al. (2006) has shown that quantities found via higher moments of the distribution function such as the heat flux require lattices with a greater number of lattice speeds $\check{\xi}$. Figure **2.2.3** depicts a two such non-standard lattice nodes. In the D2Q13 example, distributions such as 9 jump not to the next lattice node, but to the node after the next node. It might be anticipated that the implementation of boundary conditions for such lattices would be a complex affair, as will become clear in a later section.

:1



D2Q13 D2V17

Figure 2.2.3 Non-Standard Lattices for Higher Moments of the Distribution Function

Based on this work, Shan and Chen (2007) proposed an expansion into multiple relaxation time models so that the transport properties can be determined independently.

2.2.4 *Double Distribution Thermal Models*

He et al. (1998) originated an alternate approach to constructing thermal lattice Bolzmann models which is also rooted in kinetic theory. Their insight was that the internal energy density $\rho\epsilon = \rho\eta RT/2 = \frac{1}{2}\int(\boldsymbol{\xi} - \boldsymbol{u})^2 f d\boldsymbol{\xi}$ could be used to define a new distribution function g representing the internal energy:

$$g = \frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2} f \tag{2.20}$$

and thus

$$\rho\epsilon = \frac{\rho\eta RT}{2} = \int g d\boldsymbol{\xi} \tag{2.21}$$

is the zeroth moment and

$$\boldsymbol{q} = \int \boldsymbol{c} g d\boldsymbol{\xi} \tag{2.22}$$

gives the heat flux as the first moment. Using the Boltzmann equation with the full collision operator $\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} = \Omega_B$ along with **2.20** and introducing a BGK-style collision operator

$$\frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2}\Omega_B = \frac{1}{\theta_e}(g^{EQ} - g) \tag{2.23}$$

where θ_e is an 'energetic' or thermal relaxation time which is manifested macroscopically as the thermal conductivity and diffusivity, and

$$g^{EQ} = \frac{(\boldsymbol{\xi} - \boldsymbol{u})^2}{2} f^{EQ} = \frac{\rho(\boldsymbol{\xi} - \boldsymbol{u})^2}{2(2\pi RT)^{\eta/2}} e^{\frac{-(\boldsymbol{\xi}-\boldsymbol{u})^2}{2RT}} \tag{2.24}$$

it can be shown that g evolves according to a Boltzmann-like equation

$$\frac{\partial g}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial g}{\partial \boldsymbol{x}} = \frac{1}{\theta_e}(g^{EQ} - g) - \Lambda \tag{2.25}$$

where Λ is a term representing effects on conduction, compression work, and viscous dissipation. The modeling philosophy represented by equations **2.20** through **2.25** form the basis from which many lattice Boltzmann thermal models can be derived. All of these models have in common that on a single lattice, one distribution function f_i is used to model momentum while the distribution function, g_i in the original case, models thermal physics. Advantages of this approach are the simplicity of the lattice and the ability to tune the Prandtl number $Pr = \frac{\nu}{\alpha}$ by setting θ and θ_e (or their non-dimensional equivalents τ and τ_e) appropriately. A disadvantage is the doubling of the memory requirement, although a similar increase in computational overhead is common to all thermal models over their athermal counterparts.

Examples of the many double-distribution models, in addition to the original (He et al., 1998), include the model of Shi et al. (2004), in which viscous heating is handled in a simpler manner that that represented by Λ and in which the energy distribution function is guaranteed to be non-negative as is required by physics. Guo et al. (2007) developed a double distribution model which instead of using an internal energy distribution function, uses a *total* energy distribution function $\tilde{h} = \frac{\boldsymbol{\xi}^2}{2} f$ which allows for polyatomic gasses to be modeled, in contrast to the monatomic gasses treated thus far. This enables the more realistic modeling of both the constant volume and constant pressure specific heats c_v and c_p, respectively. Many practical situations are well described by the Boussinesq approximation, in which all fluid properties are assumed constant with temperature, as is density except for the influence of variable density on a buoyancy force. Guo et al. (2002) developed a double distribution function approach incorporating the Boussinesq approximation for incompressible flows using a temperature distribution function.

For many applications the viscous heating and compression work (incompressible flows) are negligible. A simplified model proposed by Peng et al. (2003) for such cases was developed by merely dropping Λ in the model of He et al. (1998) which represent these effects. Li et al. (2008) investigated the consequences of such an omission and constructed an improved simplification of He et al. (1998) which avoids these consequences.

2.2.5 *Boundary Conditions*

In contrast to CFD approaches based on the macroscopic conservation equations in which boundary conditions are given in terms of the thermohydrodynamic variables being modeled (e.g. velocity $\boldsymbol{u}$, temperature T, heat flux $\boldsymbol{q}$, etc.), boundary conditions in lattice Boltzmann methods must be stated ultimately in the form of the distributions f_i and g_i as appropriate. These boundary distributions are only partially

functions of the macroscopic variables, and the main challenge in forming boundary conditions for lattice Boltzmann methods is determining the distributions in a manner consistent with the physics being described.

For illustrative purposes, consider a D2Q9 node on a lower bounding wall as shown in figure **2.2.4**. During the operation of the lattice Boltzmann process, populations stream in from neighboring nodes in the fluid and the lower boundary. For example, distribution 7 streams in from the neighboring upper right node in the fluid and (possibly, depending on the specific implementation) collides. Thus in this situation nodes 0, 1, 3, 7, 4, and 8 are known at this boundary node, but distributions 2, 5, and 6 are unknown as there are no nodes inside the wall in this example.

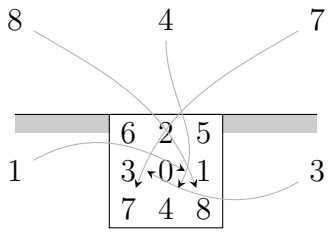


Figure 2.2.4 Node at a Floor: Distributions 2, 6, and 5 Are Unknown

The simplest method of determining these unknown (momentum) distributions (f) is the bounceback condition for no slip at a boundary. A holdover from lattice gas automata (D'Humières and Lallemand, 1987), this condition merely states that the incoming distributions are reflected back in the direction they come from, hence $f_2 = f_4$ and thus f_4 streams back into the fluid, etc. This essentially Dirichlet condition is very simple to implement and is one reason for the popularity of lattice Boltzmann methods in recent years: complex boundaries such as those in porous or granular media can be easily modeled. In studying flows in particulate suspensions, Ladd (1994b, 1994a) considered bounceback with the boundary not on nodes themselves but on the intersecting links between nodes, allowing for higher resolution in the representation of the boundary geometry. A systematic study of the bounceback

condition was conducted by He et al. (1997), finding this to be a first order method due to the presence of a slip velocity at the wall, in contrast to the effective second order nature of the lattice Boltzmann model in the fluid. The 'halfway bounceback' with the boundary considered to be on links halfway between nodes was studied as well and was shown to be second order accurate. Bounceback conditions were likewise investigated by Noble et al. (1995a), who proposed an alternative second order accurate scheme which uses nodes placed within the wall in addition to those on the boundary between the wall and fluid; these interior wall nodes supply distributions to the boundary/fluid interface.

Given that the bounceback condition leads to a slip velocity at a boundary which is fictitious in the macroscopic limit (Knudsen number $Kn \ll 1$, i.e. the microscopic length scale $\ll$ the length scale of the flow), Inamuro et al. (1995) offered a slip-canceling boundary condition in which the unknown distributions are assumed to be equilibrium distributions which incorporate 'counter slip' velocity and density terms. These slip terms are found by enforcing constraints formed using the equations for the macroscopic moments (e.g. $\boldsymbol{u}$) of the distribution functions.

Chen et al. (1996), noting that the lattice Boltzmann method is a finite-difference discretization of the Boltzmann equation, developed an extrapolation boundary condition similar to those used in traditional finite difference methods. Here, like Noble et al. (1995a), nodes are defined within the wall in addition to on the boundary, except that the distributions at these interior nodes are found by extrapolating from the nodes on the boundary and one lattice link into the fluid. All distributions then stream and collision happens everywhere except the boundary where the equilibrium distributions are used to enforce velocity conditions.

By incorporating the equations for the moments of the particle distribution function and assuming that the *non-equilibrium* portion $f^{neq} = f - f^{eq}$ of the particle distribution normal to a boundary bounces back, Zou and He (1997) developed a

second order method. The idea of the non-equilibrium distribution bouncing back was subsequently explained in greater detail and put on a more fundamental theoretical basis for both athermal and thermal (double-distribution) models by He et al. (1998). Continuing with the use of the non-equilibrium distribution, Guo et al. (2002a, 2002b) proposed a second order condition by extrapolating f^{neq} from nodes in the fluid (rather than from any nodes placed within the wall in the manner of Chen et al. (1996)), and applied this to athermal and (double-distribution) thermal models (Guo et al., 2002, 2007)

Starting from a more fundamental basis in kinetic theory (e.g. (Cercignani, 1988)), Ansumali and V. Karlin (2002) developed a "kinetic" boundary condition, which was subsequently used in thermal and rarefied flow ($Kn \approx O(1)$ or greater) regimes (Prasianakis, 2008)

All of these conditions so far have been of the Dirichlet type. D'Orazio et al. (2004) presented both Dirichlet and Neumann types using the original double distribution model of He et al. (1998). For momentum the non-equilibrium bounceback method was used while the conditions for internal energy, both in the form of a specified temperature (Dirichlet type) or specified heat flux (Neumann type), were found using a method derived from the counter-slip idea of Inamuro et al. (1995). In this thermal case, the counter slip quantity is a 'counter slip energy density' incorporated into g^{eq}; specific conditions are constructed using this g^{eq} and constraints formed from the discrete equivalents of equations **2.21** and **2.22**.

All of these conditions were originally applied onto standard lattices such as D2Q9. Non-standard lattices such as those in figure **2.2.3** will require complex boundary condition implementation, as not only does a boundary node contain unknown distributions, but nodes at least one lattice link away from the boundary will also contain unknown distributions.

Boundary conditions can also be applied to fluid nodes that are boundaries of the

lattice Boltzmann domain, for example periodic, inlet, or outlet boundary conditions in velocity or pressure as appropriate. Many of the boundary condition references listed above contain their own implementations of these conditions. However with the exception of periodic conditions, these types of boundaries are not used in this work and will not be mentioned in any detail here.

2.2.6 *Conjugate Heat Transfer with Lattice Boltzmann*

In addition to solving problems of purely fluid flow, the lattice Boltzmann technique has also been employed in conjugate simulations involving multiple heat transfer processes in multiple media, solid and fluid. The work of Wang et al. (2007) and Meng et al. (2008) used double-distribution lattice Boltzmann models to simulate transient flow and heat transfer in both thick conducting walls and in the flow domain bounded by those walls as shown in figure **2.2.5**. For the solid walls, the problem being solved is implicitly one of fluid conduction as the boundary conditions for the walls and the lack of a buoyancy force in the lattice Boltzmann model for the 'fluid' which constitutes the walls leads to a null velocity field. In the former work (Wang et al., 2007), comparisons with simulations of the same transient conjugate problem done in the commercial code FLUENT show that the lattice Boltzmann technique yields grid-converged solutions with coarser grids and in less than half the computational time.

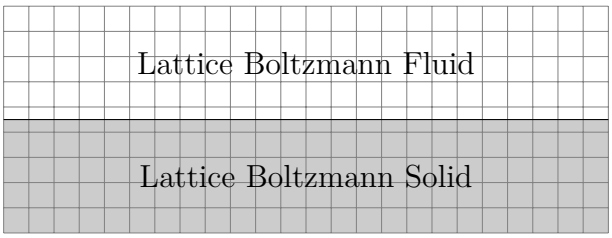


Figure 2.2.5 Conjugate Lattice Boltzmann Model for Wall Conduction and Fluid Flow

Radiation and convection across a divided enclosure was considered by Mezrhab et al. (2007) in which a finite difference solution to radiation from walls was coupled to a lattice Boltzmann model of convective flow. More recent investigations have used lattice Boltzmann models for a statically conductive and radiatively participating medium with a finite volume model of radiation from the walls bounding a cavity (Mondal and Mishra, 2009).

2.2.7 *A Note on Turbulence Modeling*

Although the modeling of turbulence is out of scope for this work, any realistic CFD model of the flows within or around buildings must include some representation of turbulence. One of the characteristic features of turbulent flows is the large spectrum of length scales (eddy sizes) present, with the largest scale being given, for example, by the geometry of the boundaries enclosing the flow, down to the very small Kolmogorov scale representing the length over which energy is viscously dissipated. In principle any CFD method could be used to simulate all scales of the flow by using a grid spacing small enough to capture the Kolmogorov scale. Such a direct numerical simulation (DNS) technique has been used with lattice Boltzmann models, e.g. see references in (Chen and Doolen, 1998 and Yu et al., 2003), however the computational overhead required for DNS is so overwhelming that this technique is limited to simulations of model flows supporting fundamental research in turbulence.

Practical problems of engineering interest employ turbulence models to capture essential features of turbulent flows without having to represent all length scales. In the context of Navier-Stokes modeling of fluid flows, the two dominant techniques are Reynolds averaged Navier-Stokes (RANS) and large-eddy-simulation (LES). In the former the velocity field is decomposed into a mean and fluctuating part, which when substituted back into the Navier-Stokes equations, leads to a term called the Reynolds stress which represents stresses due to the fluctuating component. This

extra stress term gives rise to the closure problem, in which there are more unknowns than equations. Various relatively *ad hoc* models then provide closure by modeling this stress in terms of the macroscopic flow. LES is similar in spirit, however the largest scales of the turbulent fluctuations are explicitly modeled, while a subgrid-scale model represents the flow physics at length scales smaller than the 'filter width'; hence LES represents a middle ground between RANS and DNS (Pope, 2000).

In the context of lattice Boltzmann models incorporating turbulence, most attempts have been based on LES techniques incorporating multiple relaxation times, for example simulations of jets (Yu and Girimaji, 2005) and combustion processes (Yu, 2004), flows in a building atrium with staircases (van Treeck et al., 2006), and lid-driven cavity flow (Chen, 2009).

There are some early indications that the kinetic approach to fluid flow can provide greater insight into turbulence and perhaps have some advantages in turbulence modeling. The entropic lattice Boltzmann technique (Karlin et al., 1999 and Ansumali and V. Karlin, 2002) was examined by Ansumali (2004) and shown to have a (perhaps very basic) implicitly built-in subgrid scale model; simulations of flow past a square cylinder showed good agreement with experiment for flow features associated with turbulence. Double-distribution thermal lattice Boltzmann simulations of natural convection in a square cavity by Dixit and Babu (2006) also compared well to benchmark results of turbulent convection despite not having any turbulence model explicitly built in.

Such results are surprising, and although theoretical investigations are incomplete and ongoing (Chen et al., 2004, Ansumali et al., 2004, Succi et al., 2006 and Girimaji, 2007), early indications give hope that the kinetic approach may provide improved models of turbulence.

2.3 Discussion

In this chapter, techniques for coupling CFD and heat transfer/ES simulation approaches have been discussed. It is here proposed to offer another method in which separate models of CFD and heat transfer are coupled under the umbrella of a common modeling language (Modelica) instead of coupling separate programs. The manner of this coupling will allow for the conflation of multidimensional CFD with lower dimensional processes, for example one dimensional heat conduction, given that high-resolution and multidimensional approaches for heat transfer through walls is not always necessary.

Given that the coupling is between separate models using well defined interfaces enabled by the design of the modeling language itself and not between separate programs leads to the possibility that the separate models can be constructed without the overlap seen in the coupling of separate programs, and without the need for the explicit coding of iteration between different models to achieve a consistent solution. The coupling paradigm is thus conjugate in a sense; the terms internal and external are not helpful when using a hierarchical modeling language instead of programs. This leads to the coupling *method* being the use of T and Q as the coupling variables, and the coupling *strategy* being full dynamic. In the context of the previous coupling *paradigms*, this overall approach is something of a middle road or hybrid one, being a coupling between two dimensional CFD and one-dimensional conduction yet is nevertheless conjugate in spirit because there exist simultaneous equations for all processes in one greater model. Hence it is referred to as semi-conjugate, or in the whimsical spirit of the ping-pong and onion nomenclature, as "tangelo".

Furthermore, an alternative CFD approach based on kinetic theory – lattice Boltzmann – has been introduced. This approach offers the benefits of a fairly simple computational scheme for unsteady flows and includes as an inherent part of the solution not only the velocity vector and temperature fields, but also the heat flux (and

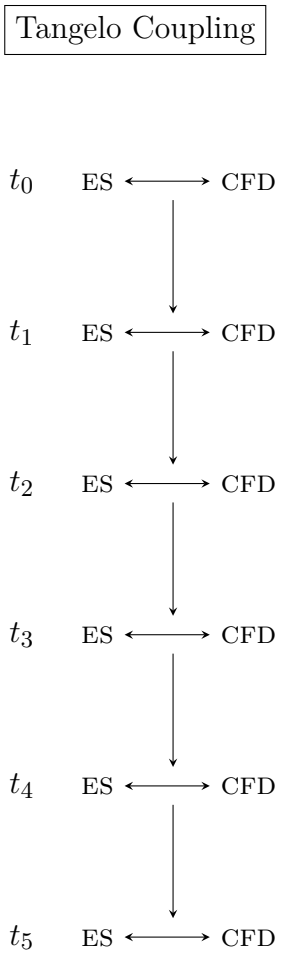


Figure 2.3.1 Tangelo Coupling

thus heat flow rate). Heat flux is just another moment of the distribution function and not a derived quantity: there is no need to compute the derivative of the temperature field and the calculation of heat flux is local to a node. Indeed, although the term "Neumann" has been used above to refer to the application of a specified heat flux in lattice Boltzmann models, this term does not truly apply here because a gradient is not being specified. In the context of Boltzmann based approaches, the heat flux is of a similar character to velocity – both are moments of a distribution function, and in the context of double distribution models they are the same order moment – and the application of a heat flux at a boundary is conceptually no different than the application of a velocity at a boundary.

The combination of the coupling approach outlined above with the lattice Boltzmann method suggest that a simple, straightforward approach is possible, using as the coupling variables the actual ones of interest, Q and T, without the need for any intermediate variables such as h.

CHAPTER 3: IMPLEMENTATION

3.1 Description of the Lattice Boltzmann Model Used

The specific lattice Boltzmann model used in this work is the two dimensional, nine molecular velocity double distribution model of Li et al. (2008) which is a simplification of the original double distribution model of He et al. (1998). This simplified model excludes viscous dissipation and compression work, both of which are negligible in the low Brinkmann and Mach number airflows associated with buildings; hence the term Λ, along with the requirement to compute temporal and spatial derivatives of the velocity field, is unnecessary. However Λ is not simply dropped in the manner of Peng et al. (2003); rather a modified model is created for both f_i and g_i. A brief description of the model follows.

For momentum the incompressible scheme of Guo et al. (2000) is used, along with the Boussinesq approximation of fluid properties being constant with temperature with the exception of density, and even then only in the context of a buoyancy force. Here the buoyancy force is expressed as $\boldsymbol{G} = \rho_0 \boldsymbol{g} - \rho_0 \boldsymbol{g} \beta (T - T_m)$; ρ_0 is a reference density, in this work taken to be the density of air at standard temperature T_0 and pressure p_0, 293.15 K and 101.325 kPa, respectively; T_m is a 'reference' temperature, often taken to be the average of the temperature field; $\boldsymbol{g}$ is the gravity vector; β is the coefficient of thermal expansion, taken here to be that of an ideal gas which is to say that $\beta = 1/T_0$. This form can be simplified by absorbing the static term $\rho_0 \boldsymbol{g}$ into the pressure so that hereafter $\boldsymbol{G} = -\rho_0 \boldsymbol{g} \beta (T - T_0)$. The evolution equation is the standard one with the inclusion of the buoyancy force

$$f_i(\boldsymbol{x} + \boldsymbol{\xi}_i \delta t, t + \delta t) = f_i(\boldsymbol{x}, t) - \frac{1}{\tau}\left[f_i(\boldsymbol{x}, t) - f_i^{eq}(\boldsymbol{x}, t) \right] + \delta t \boldsymbol{G} \cdot \frac{\check{\xi}_i - \boldsymbol{u}}{p} f_i^{eq} \qquad (3.1)$$

whereas the equilibrium distributions f^{eq} are given by

$$f_i^{eq} = \begin{cases} \rho_0 - \frac{20}{12}\frac{p}{\check{\xi}^2} + \rho_0 \tilde{f}_i^{eq}, & i = 0 \\ \frac{1}{3}\frac{p}{\check{\xi}^2} + \rho_0 \tilde{f}_i^{eq}, & i = 1,2,3,4 \\ \frac{1}{12}\frac{p}{\check{\xi}^2} + \rho_0 \tilde{f}_i^{eq}, & i = 5,6,7,8 \end{cases} \tag{3.2}$$

$\tilde{f}_i^{eq}$ is

$$\tilde{f}_i^{eq} = \rho_0 w_i \left[\frac{3}{\check{\xi}^2}(\boldsymbol{\xi}_i \cdot \boldsymbol{u}) + \frac{9}{2}\frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{\check{\xi}^4} - \frac{3}{2}\frac{(\boldsymbol{u} \cdot \boldsymbol{u})}{\check{\xi}^2} \right] \tag{3.3}$$

and the weights w_i are the same as before

$$w_i = \begin{cases} \frac{4}{9}, & i = 0 \\ \frac{1}{9}, & i = 1,2,3,4 \\ \frac{1}{36}, & i = 5,6,7,8 \end{cases} \tag{3.4}$$

In contrast to more typical lattice Boltzmann models, the pressure rather than the density is the (modified) zeroth moment of f_i.

$$p = \frac{12\check{\xi}^2}{20}\left[\sum_{i=1}^{8} f_i + \rho_0 \tilde{f}_0^{eq} \right] \tag{3.5}$$

The velocity remains the familiar first moment

$$\boldsymbol{u} = \frac{1}{\rho_0}\sum_{i=0}^{8} \boldsymbol{\xi}_i f_i \tag{3.6}$$

For the internal energy distributions g_i, a new equilibrium form g_i^{eq} is determined which allows for the proper exclusion of viscous heating and compression work terms. In the context of the Boussinesq approximation this is

$$g_i^{eq} = w_i \rho_0 \frac{\eta}{2} RT \left[1 + 3\frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})}{\check{\xi}^2} + \frac{9}{2}\frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{\check{\xi}^4} - \frac{3}{2}\frac{(\boldsymbol{u} \cdot \boldsymbol{u})}{\check{\xi}^2} \right] \tag{3.7}$$

The evolution of g_i is given as

$$g_i(\boldsymbol{x} + \boldsymbol{\xi}_i \delta t, t + \delta t) = g_i(\boldsymbol{x}, t) - \frac{1}{\tau_e}\left[g_i(\boldsymbol{x}, t) - g_i^{eq}(\boldsymbol{x}, t) \right] \tag{3.8}$$

however it will be shown in the next chapter that this discretization is insufficient to recover the heat flux at a greater than $O(1)$ rate of grid convergence. For this the discretization of the original double distribution scheme will be necessary (He et al., 1998) and will be detailed in Chapter 4.

A Chapman-Enskog expansion reveals that this scheme recovers the correct macroscopic equations for the velocity and temperature fields and that the kinematic viscosity ν and thermal diffusivity α are given by

$$\begin{aligned} \nu &= \left(\tau - \frac{1}{2}\right)\frac{\check{\xi}^2\delta t}{3} \\ \alpha &= \left(\tau_e - \frac{1}{2}\right)\frac{\check{\xi}^2\delta t}{3} \end{aligned} \tag{3.9}$$

as shown in (Guo et al., 2000 and Li et al., 2008). It can be easily shown that while the viscosity and thermal diffusivity for air can be correctly specified – and hence so can the Prandtl number $Pr = \nu/\alpha$ – the thermal conductivity κ and specific heat c_p as given by this model cannot be. While lattice Boltzmann models exist which can overcome this deficiency, they are more complex and are mainly relevant for the modeling of acoustic physics, which is not of interest here. Furthermore, results in the next chapter will demonstrate that the correct heat flux can be had by re-scaling by the ratio of the thermal conductivity of the model to that of air.

For many double distribution thermal models, it is essential that the lattice velocity $\check{\xi}_i = \delta x/\delta t$ also be equal to the mean molecular speed $\sqrt{3RT}$ for correct physics. Fortunately this is not the case in this model (Li et al., 2008) as this would dictate very small timesteps.

The moments of g_i then give the temperature and heat flux

$$T = \frac{1}{\rho_0 R}\sum_{i=0}^{8} g_i \tag{3.10}$$

$$\boldsymbol{q} = \sum_{i=0}^{8} \boldsymbol{c}_i g_i \tag{3.11}$$

with $\boldsymbol{c}_i = \boldsymbol{\xi}_i - \boldsymbol{u}$ being the peculiar velocity.

The boundary condition used for the momentum distributions f_i was the non-equilibrium extrapolation scheme since it works well the moving boundaries seen in the planar Couette cases considered in Chapter 4 and is second-order accurate (Guo et al., 2002a). The basic idea of this condition is that the distributions at a wall can be decomposed into equilibrium and non-equilibrium parts $f_i(\boldsymbol{x}_{wall}, t) = f_i^{eq}(\boldsymbol{x}_{wall}, \boldsymbol{u}, p, t) + f_i^{neq}(\boldsymbol{x}_{wall}, t)$ and that these two components can be determined by extrapolation from the known distributions at nodes neighboring the wall. The equilibrium component is partially known as the velocity at the wall is typically known, however the pressure is not; in the manner of Li et al. (2008)

$$f_i^{eq}(\boldsymbol{x}_{wall}, \boldsymbol{u}, p, t) \approx \begin{cases} \frac{1}{3}\frac{p(\boldsymbol{x}_f,t)}{\check{\xi}^2} + \rho_0 \tilde{f}_i^{eq}(\boldsymbol{x}_{wall}, t), & i = 1,2,3,4 \\ \frac{1}{12}\frac{p(\boldsymbol{x}_f,t)}{\check{\xi}^2} + \rho_0 \tilde{f}_i^{eq}(\boldsymbol{x}_{wall}, t), & i = 5,6,7,8 \end{cases} \tag{3.12}$$

where $\boldsymbol{x}_f$ denotes the nearest node in the fluid, away from the wall. The non-equilibrium component is estimated as

$$f_i^{neq}(\boldsymbol{x}_{wall}, t) \approx f_i(\boldsymbol{x}_f, t) - f_i^{eq}\boldsymbol{x}_f, t \tag{3.13}$$

The boundary condition for the internal energy distributions g_i was taken either as the counter-slip approach of Inamuro et al. (1995) as modified by D'Orazio et al. (2004), or as the non-equilibrium bounceback condition of Zou and He (1997), as described by He et al. (1998). The counter-slip condition assumes that the unknown g_i are the equilibrium distributions with a 'counter-slip' internal energy density $\acute{\epsilon}$, the non-slip internal energy density being $\epsilon = \frac{\eta}{2}RT$

$$g_i^{unknown} = \rho_0(\epsilon_{wall} + \acute{\epsilon})w_i\left[1 + 3\frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})}{\check{\xi}^2} + \frac{9}{2}\frac{(\boldsymbol{\xi}_i \cdot \boldsymbol{u})^2}{\check{\xi}^4} - \frac{3}{2}\frac{(\boldsymbol{u} \cdot \boldsymbol{u})}{\check{\xi}^2}\right] \tag{3.14}$$

For a specified temperature at the wall T_{wall}, the constraint is

$$\sum_{i=0}^{8} g_i = \rho_0 \epsilon_{wall} = \rho_0 \frac{\eta}{2} R T_{wall} \tag{3.15}$$

Substituting equation **3.14** into equation **3.15** yields an equation for $\rho_0(\epsilon_{wall} + \acute{\epsilon})$ which is then used back in equation **3.14** for the $g_i^{unknown}$.

During the course of this work it was found that the non-equilibrium bounceback scheme for g_i is more easily implemented than the counter-slip condition while giving essentially the same results, and also led to the simple creation of Robin type boundary nodes in addition to the Dirichlet types discussed until now. This condition can be compactly given as

$$(g_\alpha - g_\alpha^{eq}) - \boldsymbol{\xi}_\alpha (f_\alpha - f_\alpha^{eq}) = \boldsymbol{\xi}_\beta (f_\beta - f_\beta^{eq}) - (g_\beta - g_\beta^{eq}) \tag{3.16}$$

Here, α and β indicate opposite molecular velocities, for example if $\alpha = 6$, then $\beta = 8$ (He et al., 1998). This type of condition can easily handle Dirichlet (specified temperature) conditions via the equilibrium distributions g_i^{eq}. Robin type boundary conditions can be defined with the same non-equilibrium bounceback scheme given connectors at these nodes for temperature and heat flow rate as a `flow` variable as it and not the heat flux is the conserved quantity in the general case; a solution can be found if this Robin node is connected to another model with sufficient equations relating temperature and heat flow rate. The heat flow rate at a node – in W/m in this two-dimensional model – is calculated from the heat flux at a node by multiplying the heat flux by the grid spacing.

3.2 Manifestation in Modelica

The model described in the previous section was implemented in Modelica as a discrete model with the timesteps being events as created using the built-in function `sample()`. The lattice Boltzmann evolution equations written here are in terms of

the new distribution, e.g. $f_i(\boldsymbol{x} + \boldsymbol{\xi}_i\delta t, t + \delta t) = F(f_i(\boldsymbol{x}, t))$, however when using Modelica it is more convenient to transform this so that the current distribution is described in terms of the previous distribution `f_i(x) = F( pre( f_i(x - ξ_iδt) ) )`.

Modelica being a language for hierarchical model creation, the lattice Boltzmann model is built up from constituent submodels, the foundation of which are the `node` models. These models contain the evolution equations, equilibrium distributions equations, boundary condition equations as appropriate, and access the shared field variables $\boldsymbol{u}$, p, and T and distributions f_i, g_i^{eq}, etc., using implicit connections enabled by the `inner/outer` construct. Nodes are collected to form (CFD) domains, can exist in the core or on the edge of a such a CFD domain, and can represent a fluid or a wall. Nodes on the edge can be fluid or wall nodes, but always implement boundary conditions. Figure **3.2.1**) gives a schematic example of a generic domain, and a more specific example with fluid nodes (unfilled) and edge nodes (squares) implementing Dirichlet (D) or Robin (R) conditions on a wall (shaded) or periodic boundary conditions in a fluid (P).

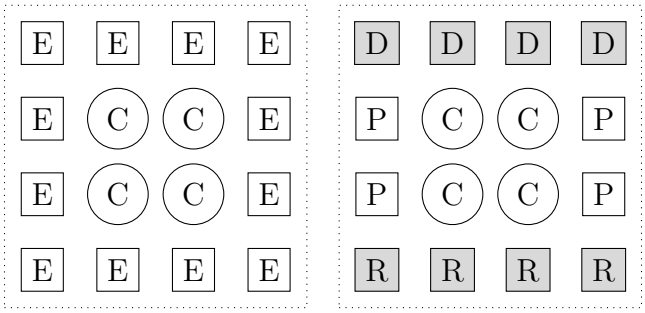


Figure 3.2.1 Collection of Nodes into CFD Domains

Domains are further incorporated into models of specific flow situations as will be shown in the next chapter. An important class of models are the boundary interfaces,

which are intermediary between the CFD domain and other models, such as conduction through a wall. In this work some of the walls are considered to be isothermal, and for these cases the interfaces are simple 'pass-throughs' which equate temperatures and sum heat flows between the nodes of a CFD domain and the bounding wall. Figure **3.2.2** gives an overview of the package structure containing relevant models, and Appendix B gives examples of the models as implemented in Modelica.

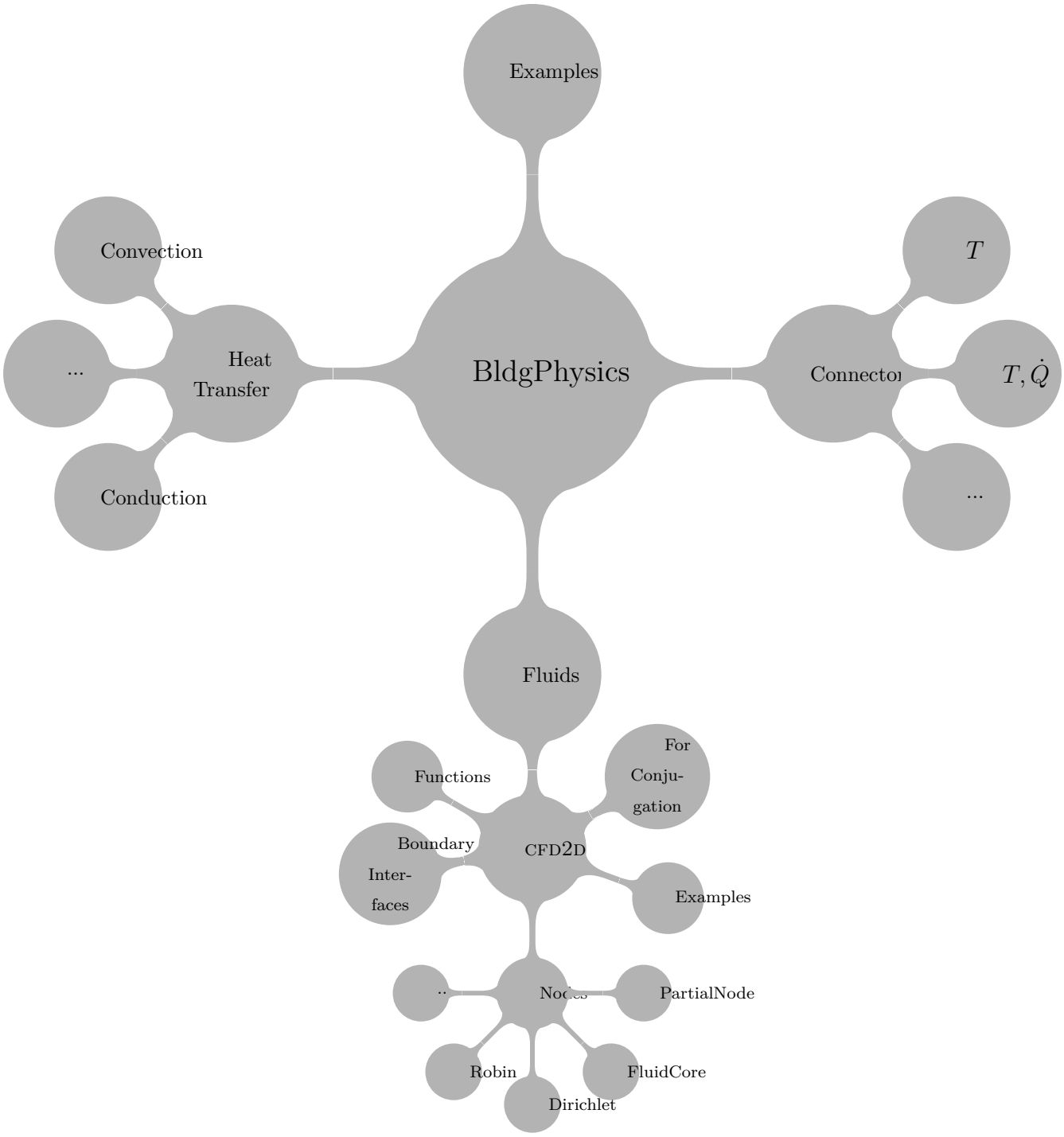


Figure 3.2.2 Abbreviated Package Structure

CHAPTER 4: STUDIES, EVALUATIONS, AND RESULTS

In this chapter the lattice Boltzmann model described in the previous chapter is used on a number of test problems which either have analytical solutions or benchmarked data in the literature. The first section deals with problems of a purely fluid-mechanical nature, while the second section considers problems to test the coupling of CFD to conductive heat transfer; some of these latter problems are typically considered to be pure fluid-mechanical ones but here been modeled here in a semi-conjugate manner.

The semantic distinctions between verification and validation here follow those of Roache (1998). Verification refers to the idea that the implementation of a model actually represents that model, i.e. the implementation is mathematically correct. In practical terms, that the computational (discretized) solution approaches the continuum solution as the grid spacing goes to zero. The rate at which this occurs implies a level of practical usefulness with respect to computational efficiency. Validation refers to the idea that the model that has been implemented is appropriate to the practical question being asked, i.e. that the (verified) equations are the correct equations to use. In practical terms, that the computational (discretized) solution describes a physical situation – that it agrees with experiment, for example. The closeness of this agreement implies a practical usefulness of a sort subtly different from that of verification in that the validation can be used as a prediction of the outcome of a physical experiment, or that the computational solution can be used as a tool for the understanding of a physical experiment.

All cases were run using version 7.4 of Dymola (Dassault Systemes, 2010), either on a Macintosh laptop running Windows XP in a virtual machine or on a desktop PC running Windows XP. Initialization was done by setting all distributions f_i and g_i

to their equilibrium values, a typical technique yet one which nevertheless introduces error in the early timesteps (Skordos, 1993).

4.1 Test Cases: Fluid Flow Only

The following cases represent situations modeled strictly as fluid-mechanics problems; none of the multi-domian capabilities of Modelica are exploited and thus represent CFD solutions only.

4.1.1 *Thermal Planar Couette*

As a first test of the implementation of a lattice Boltzmann model in Modelica, the planar Couette problem is considered for its simplicity and the availability of analytical solutions. This problem consists of two infinite parallel plates bounding a fluid of viscosity ν and thermal diffusivity $\alpha = \kappa/\rho c_p$. At time $t < 0$ the fluid is uniformly at rest and at temperature T_{bottom}, and at $t >= 0$ the upper plate is impulsively set into motion to the right at velocity u_{upper} while its temperature is instantaneously set to $T_{upper} > T_{bottom}$. The geometry is shown in figure **4.1.1**

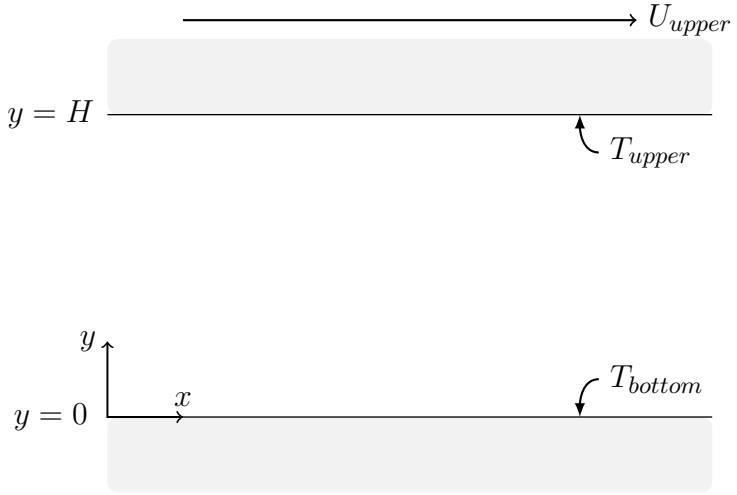


Figure 4.1.1 The Planar Couette Problem

In this situation the Navier-Stokes equations reduce to:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial y^2} \tag{4.1}$$

with boundary conditions, using the coordinate system with the origin on the bottom surface:

$$\begin{aligned} t < 0 : u(y,t) &= 0 \\ t \geq 0 : u(H,t) &= U_{upper} \\ u(0,t) &= 0 \end{aligned} \tag{4.2}$$

$$\begin{aligned} t < 0 : T(y,t) &= T_{bottom} \\ t \geq 0 : T(H,t) &= T_{upper} \\ T(0,t) &= T_{bottom} \end{aligned} \tag{4.3}$$

Because $T_{upper} > T_{bottom}$ the fluid is stably stratified and there is no vertical component of velocity, thus the thermal aspect of this problem reduces to one of fluid conduction in the y direction. Thus the heat diffusion equation reduces to a form analogous to equation **4.1**:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial y^2} \tag{4.4}$$

with boundary conditions:

$$\begin{aligned} t < 0 : T(y,t) &= T_{bottom} \\ t \geq 0 : T(H,t) &= T_{upper} \\ T(0,t) &= T_{bottom} \end{aligned} \tag{4.5}$$

Solutions to these equations can be found in many textbooks. Schlichting (1979) gives the solution to equation **4.1** as:

$$\frac{u(y \Rightarrow H-y,t)}{U_{upper}} = \sum_{n=0}^{\infty} erfc\left[2n\frac{H}{2\sqrt{\nu t}} + \frac{y}{2\sqrt{\nu t}}\right] - \sum_{n=0}^{\infty} erfc\left[2(n+1)\frac{H}{2\sqrt{\nu t}} - \frac{y}{2\sqrt{\nu t}}\right] \tag{4.6}$$

where $erfc$ is the complimentary error function, and the transformation $y \Rightarrow H-y$ is necessary as the boundary conditions in Schlichting (1979) are flipped up–down from those given here.

An alternate solution to equation **4.1** is given by White (2006), and is adapted here as the solution for equation **4.4**:

$$\frac{T(y \Rightarrow H-y,t^*) - T_{bottom}}{T_{upper} - T_{bottom}} = \left(1 - \frac{y}{h}\right) - \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}e^{-n^2\pi^2t^*}sin\left(\frac{n\pi y}{h}\right) \tag{4.7}$$

Where $t^* = \frac{(\nu t)}{H^2}$ is a dimensionless time. From this it is simple to show that the heat flux is

$$\begin{aligned} q(y \Rightarrow H-y,T,t^*) &= \kappa\frac{\partial T}{\partial y} \\ &= \kappa\left\{-\frac{\Delta T}{h} - \frac{2(\Delta T)}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}e^{-n^2\pi^2t^*}\left(\frac{n\pi}{h}\right)cos\left(\frac{n\pi y}{h}\right)\right\} \end{aligned} \tag{4.8}$$

where $\Delta T = T_{upper} - T_{bottom}$. Note that both equations **4.6** and **4.7** give linear profiles of velocity and temperature at $t \rightarrow \infty$.

The initial cases were run with the standard lattice Boltzmann discretization scheme for both the momentum and internal energy distributions, as described in the previous chapter. The boundary conditions for the momentum distributions f_i on the top and bottom walls were implemented using non-equilibrium extrapolation (Guo et al., 2002a, 2002b). The counterslip approach was taken for the implementation of Dirichlet boundary conditions for the internal energy distributions g_i on the top and

bottom walls (D'Orazio et al., 2004). Periodic boundary conditions were applied for unknown f_i and g_i on the sides. Figure **4.1.2** depicts the CFD domain used.

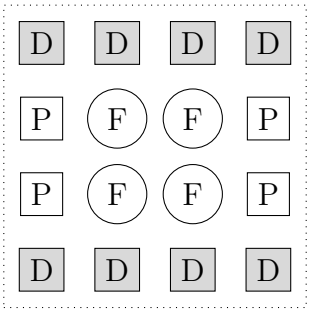


Figure 4.1.2 The Planar Couette Problem: Modelica Scheme

Criteria for momentum to considered at steady-state was that given by He et al. (1996):

$$\frac{\sum_{i,j} \parallel \boldsymbol{u}(\boldsymbol{x}_{i,j}, t) - \boldsymbol{u}(\boldsymbol{x}_{i,j}, t - \delta t) \parallel}{\sum_{i,j} \parallel \boldsymbol{u}(\boldsymbol{x}_{i,j}, t - \delta t) \parallel} < 10^{-6} \tag{4.9}$$

where δt is the timestep and $\parallel \bullet \parallel$ denotes the L_2 norm. Criteria for temperature to be considered at steady state was

$$max \left[\frac{T(\boldsymbol{x}, t) - T(\boldsymbol{x}, t - \delta t)}{T(\boldsymbol{x}, t)} \right] < 10^{-10} \tag{4.10}$$

Both criteria had to be satisfied for a simulation to be considered to have reached a steady state condition.

Using the properties of air with a temperature difference of 3 K, the lattice Boltzmann model gives unsteady results which compare very well to the analytical solutions. The evolution in normalized coordinates is given in figure **4.1.3** for the velocity and figure **4.1.4** for the temperature. For convenience and consistency, the nondimensional time is taken to be that of Schlichting (1979) for the depiction of both results.

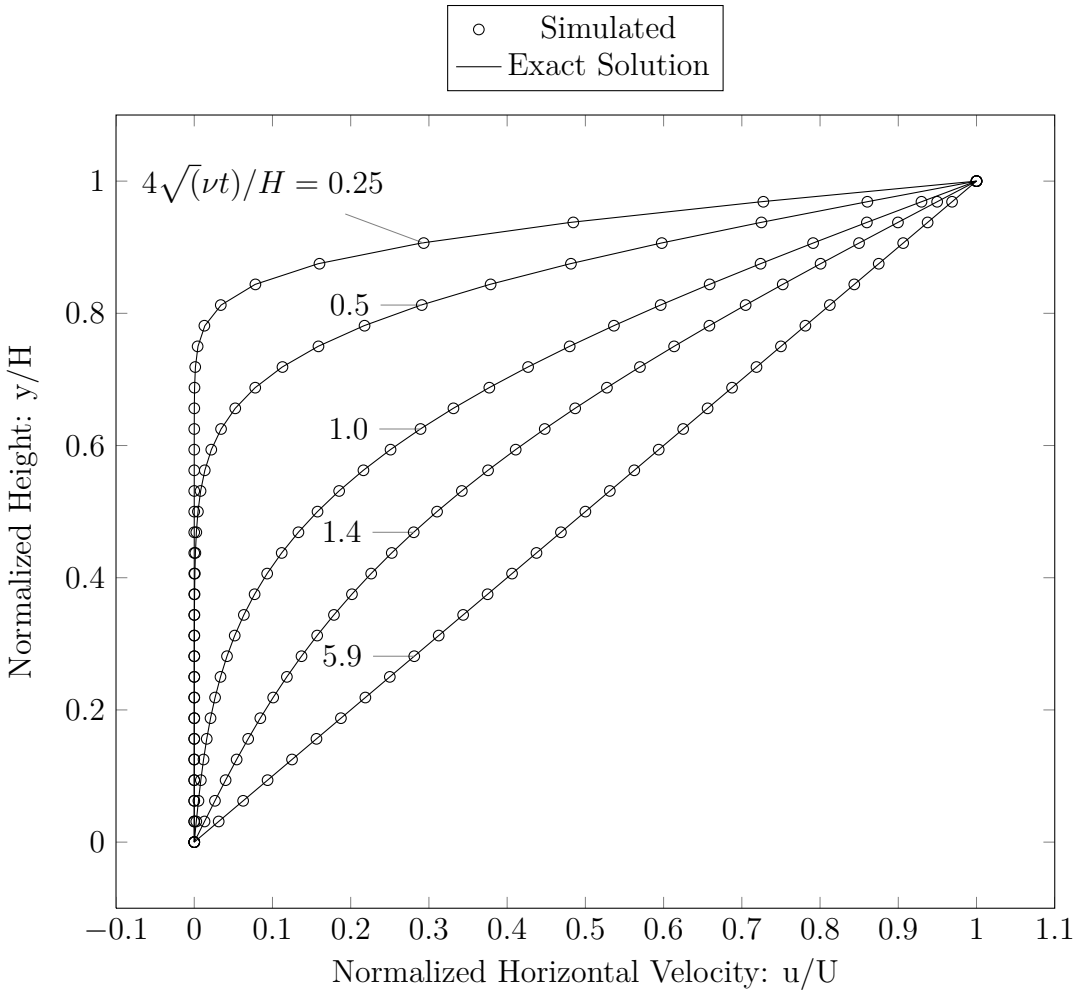


Figure 4.1.3 Planar Couette Velocity Evolution: Std. Discretization of g

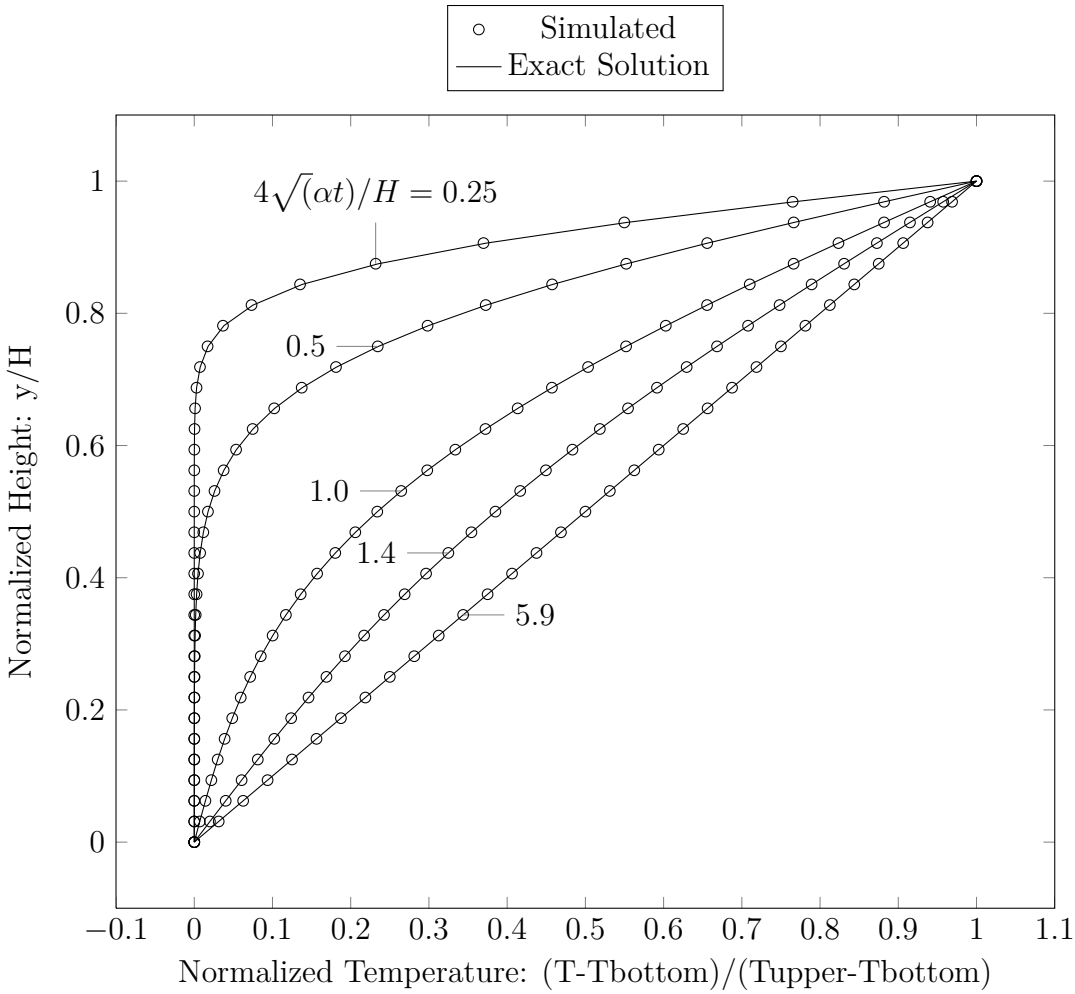


Figure 4.1.4 Planar Couette Temperature Evolution: Std. Discretization of g

Despite the fact that the temperature evolves correctly, the heat flux is does not, as shown in figure **4.1.5** for the upper surface. Furthermore, the steady state values of the heat flux do not assume the values of the exact solution. Table **4.1.1** compares the exact and simulated steady state heat fluxes for a variety of grid spacings and CFD domain heights.

It can be seen in table **4.1.1** that the simulated heat flux nevertheless approaches the exact solution as the grid spacing is reduced, suggesting some degree of verification. An assessment of the rate of spatial convergence (grid refinement study) for the heat flux was conducted using the following equation (Roache, 1998)

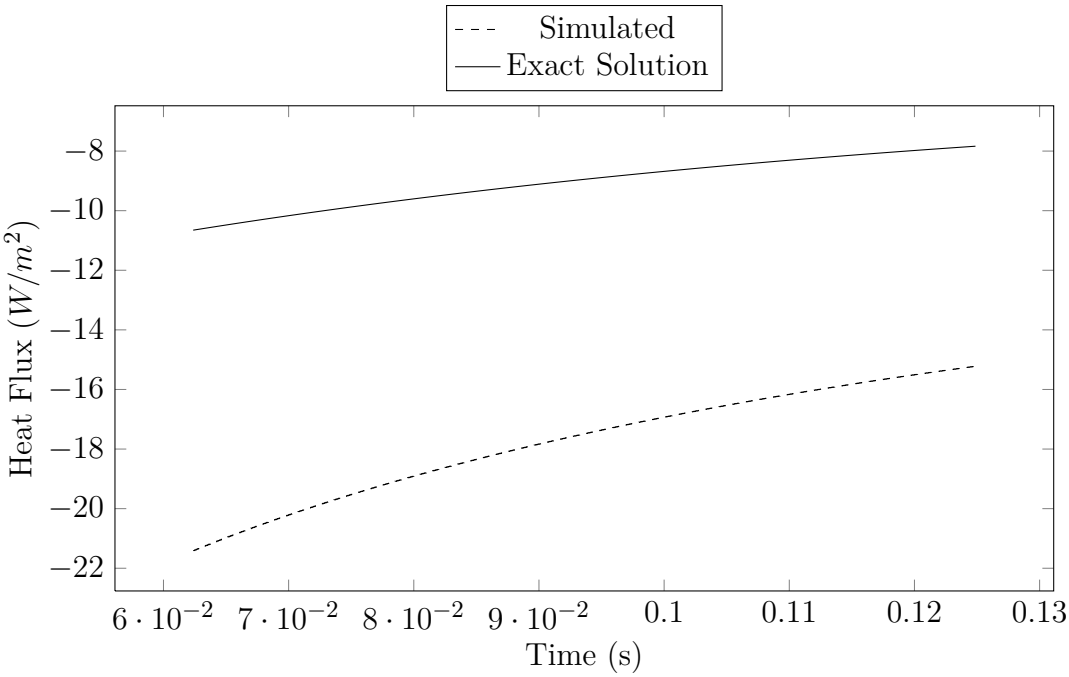


Figure 4.1.5 Planar Couette Upper Surface Heat Flux Evolution: Std. Discretization of g

$$\Upsilon = \frac{ln\left(\frac{q|_{grid3} - q|_{grid2}}{q|_{grid2} - q|_{grid1}}\right)}{ln(n)} \tag{4.11}$$

where Υ is the order of spatial convergence and n is the scaling factor between grid sizes, i.e. if the grid spacing is δx, then

$$n = \frac{\delta x|_{grid3}}{\delta x|_{grid2}} = \frac{\delta x|_{grid2}}{\delta x|_{grid1}} \tag{4.12}$$

Table **4.1.2** gives the results, indicating that this model, while showing good results in the velocity and temperature fields, only has first order grid convergence in the heat flux – the contracted second moment of the internal energy density distribution – this is a slow and impractical rate. Extrapolation from the results in table **4.1.1** shows that the exact solution is only approached linearly and hence the grid spacing must be very small.

Table 4.1.1 Planar Couette Steady State Heat Flux: Std. Discretization of g

# Rows	**Height** *(m)*	**q, simulated** W/m^2	**q, exact** W/m^2
	0.008	-24.48	-2.88
	0.004	-27.36	-5.75
9	0.002	-33.11	-11.51
	0.001	-44.62	-23.02
	0.0005	-67.64	-46.04
	0.008	-13.68	-2.88
	0.004	-16.56	-5.75
17	0.002	-22.31	-11.51
	0.001	-33.82	-23.02
	0.0005	-56.84	-46.04
	0.008	-10.08	-2.88
	0.004	-12.96	-5.75
25	0.002	-18.71	-11.51
	0.001	-30.22	-23.02
	0.0005	-53.24	-46.04
	0.008	-8.28	-2.88
	0.004	-11.16	-5.75
33	0.002	-16.91	-11.51
	0.001	-28.42	-23.02
	0.0005	-51.44	-46.04

Table 4.1.2 Planar Couette Grid Convergence of Steady State Heat Flux: Std. Discretization of g

Height *(m)*	**# Rows**	**q, simulated** W/m^2	**q, exact** W/m^2	**log(δx)**	**log(q_{sim} - q_{exact})**	Υ
	9	-33.11		-3.60	1.33	
0.002	17	-22.31	-11.51	-3.90	1.03	1.00
	33	-16.9105		-4.20	0.7324	

In order to improve this situation, the standard discretization scheme of Li et

al. (2008) is abandoned in favor of the scheme of He et al. (1998) in the original double distribution model. In this original model a true second order discretization was employed to reconcile the conflict between the viscous dissipation's dependance on the viscosity as expressed by purely kinetic theory considerations, θRT, and the requirement that the viscosity must be expressed in modified form as $(\theta - 1/2\delta t)RT$ in order to eliminate a truncation error (the original double distribution model requires that $\check{\xi} = \sqrt{3RT}$, hence the equation relating viscosity to the relaxation time being different from that in Chapter 3). The model of Li et al. (2008) used here does not include viscous heating, and so this inconsistency does not arise and the standard discretization can be used, except, as shown above, when the heat flux is to be computed not as a derivative of the temperature field but as a byproduct of the distributions g_i.

The second order scheme introduced by He et al. (1998) converts the Boltzmann equation for the internal energy distribution – equation **2.25** without the term Λ – into an implicit evolution equation

$$\begin{aligned} g(\boldsymbol{x}+\boldsymbol{\xi}\delta t,\boldsymbol{\xi},t+\delta t) - g(\boldsymbol{x},\boldsymbol{\xi},t) = &-\frac{1}{2\tau_e}\big[g(\boldsymbol{x}+\boldsymbol{\xi}\delta t,\boldsymbol{\xi},t+\delta t) - g^{eq}(\boldsymbol{x}+\boldsymbol{\xi}\delta t,\boldsymbol{\xi},t+\delta t)\big] \\ &-\frac{1}{2\tau_e}\big[g(\boldsymbol{x},\boldsymbol{\xi},t) - g^{eq}(\boldsymbol{x},\boldsymbol{\xi},t)\big] \end{aligned} \tag{4.13}$$

This equation can be made explicit by the substitution

$$gm = g + \frac{1}{2\tau_e}(g - g^{eq}) \tag{4.14}$$

so that equation **4.13** becomes

$$gm(\boldsymbol{x}+\boldsymbol{\xi}\delta t,\boldsymbol{\xi},t+\delta t) - gm(\boldsymbol{x},\boldsymbol{\xi},t) = -\frac{1}{\tau_e+1/2}\big[gm(\boldsymbol{x},\boldsymbol{\xi},t) - g^{eq}(\boldsymbol{x},\boldsymbol{\xi},t)\big] \tag{4.15}$$

As the same lattice can be used for both the momentum distributions and the internal energy distributions, equation **4.15** is fully discretized in temporal, physical, and

velocity spaces by substituting $\boldsymbol{\xi}_i$ for $\boldsymbol{\xi}$. The temperature can be shown to be the same as before:

$$T = \frac{1}{\rho_0 R} \sum_{i=0}^{8} g m_i \tag{4.16}$$

However the equation for the heat flux must be modified in light of of equation **4.14**:

$$\boldsymbol{q} = \frac{\tau_e}{\tau_e + 1/2} \Big[\sum_{i=0}^{8} \boldsymbol{c}_i g m_i - \rho_0 \epsilon \boldsymbol{u} \Big] \tag{4.17}$$

Once this modification is made and the model of planar Couette flow is run again, the evolution of velocity and temperature still tracks the exact solution as shown in figures **4.1.6** and **4.1.7**

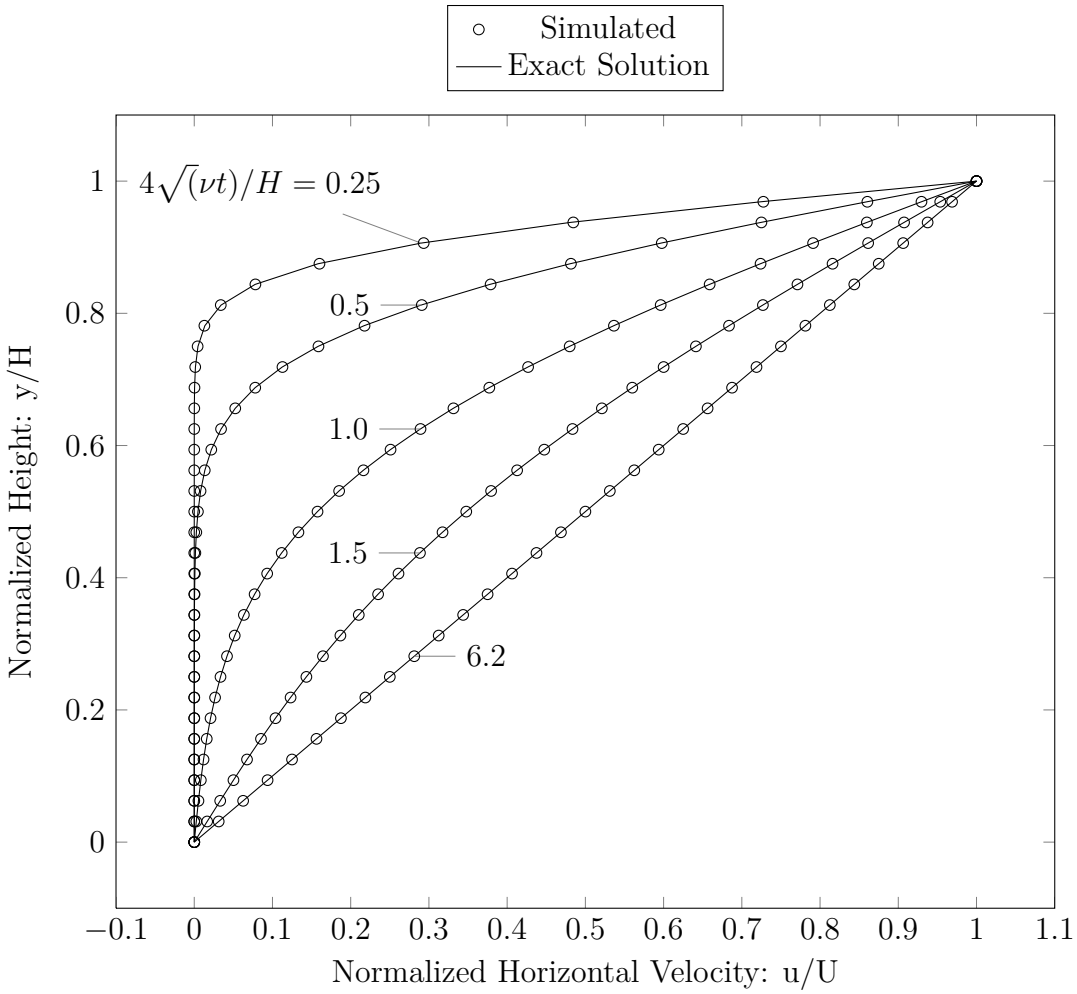


Figure 4.1.6 Planar Couette Velocity Evolution: 2nd Order Discretization of g

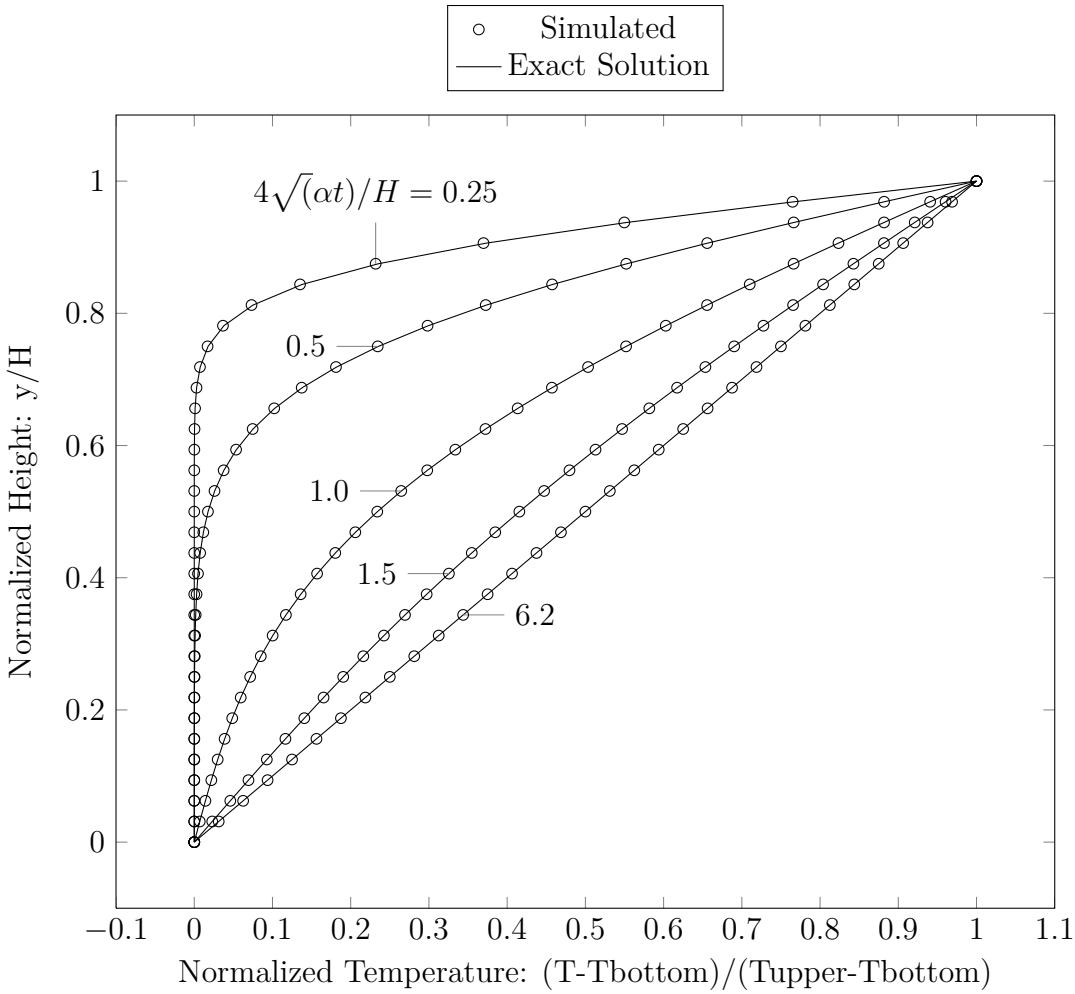
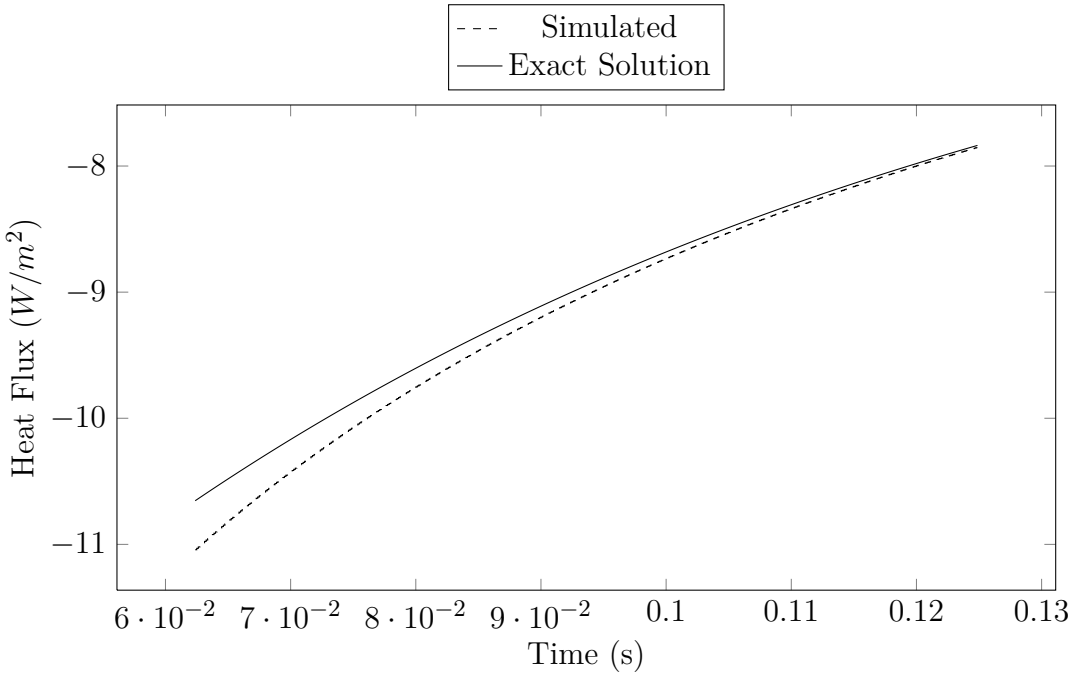


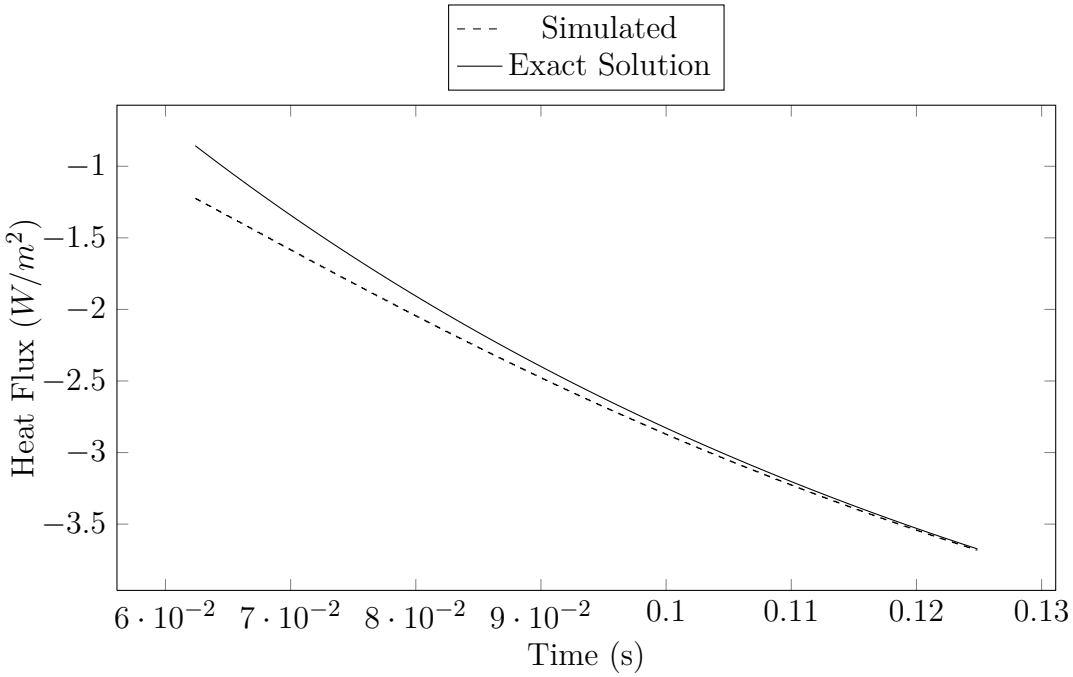
Figure 4.1.7 Planar Couette Temperature Evolution: 2nd Order Discretization of g

However now the simulated heat flux's evolution tracks much more closely to the exact solution as shown in figure **4.1.8**.

In Chapter 3 it was pointed out that the thermal diffusivity α may be correctly specified for air, but that the thermal conductivity cannot be. Hence the results given thus far are for the thermal conductivity of the model, which in this case is $\kappa_{model} = 0.00767\ \frac{W}{mK}$ in contrast to that of air at $\kappa_{air} = 0.02572\ \frac{W}{mK}$ at standard temperature and pressure. However this can be overcome by rescaling $\boldsymbol{q}_{air} = (\kappa_{air}/\kappa_{model})\boldsymbol{q}_{model}$. Figure **4.1.9** shows the evolution of the simulated heat flux compared to the exact



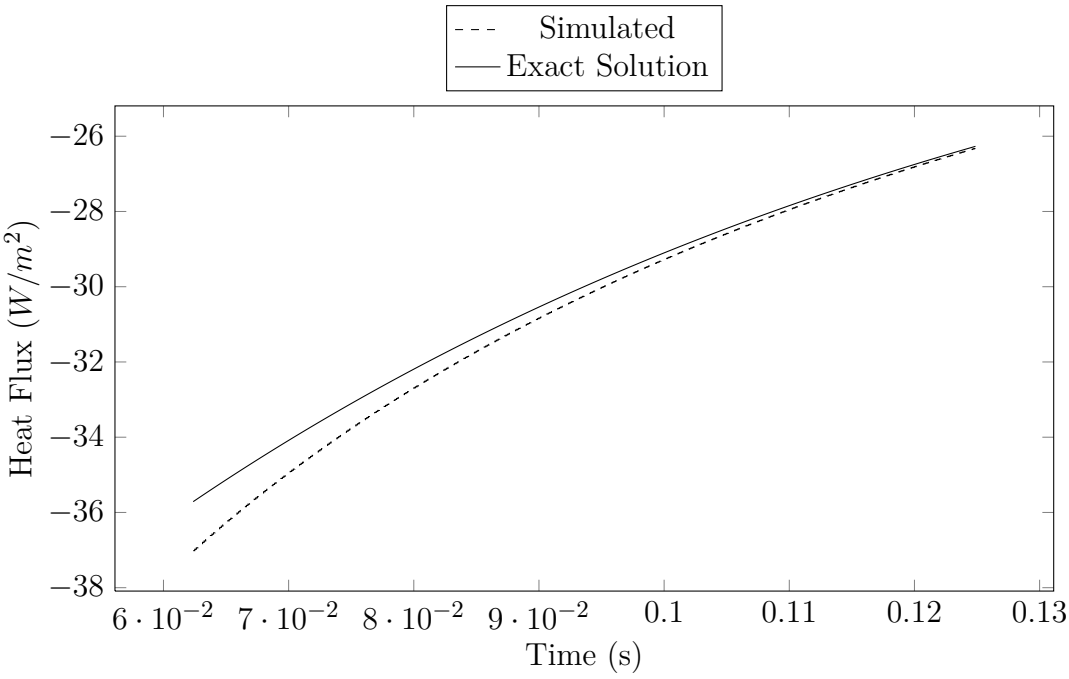
Upper Surface



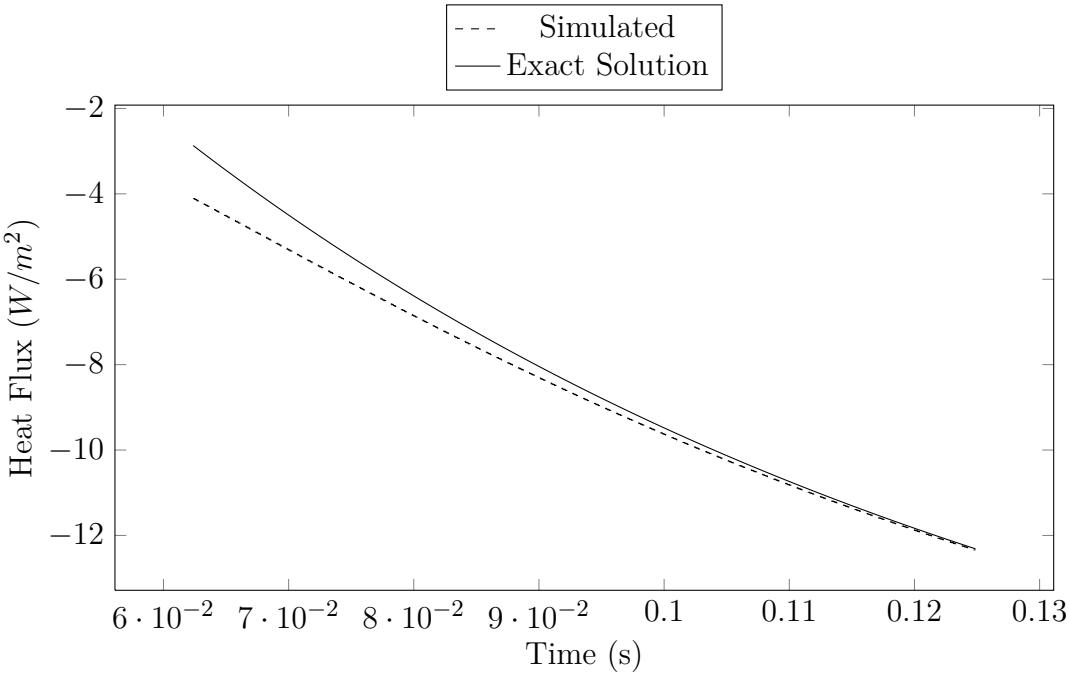
Bottom Surface

Figure 4.1.8 Planar Couette Surface Heat Flux Evolution: 2^{nd} Order Discretization of g

solution found using κ_{air}; it it believed that the discrepancy early in the plots are a reflection of the initialization of the distributions with their equilibrium values.



Upper Surface



Bottom Surface

Figure 4.1.9 Planar Couette Surface Heat Flux Evolution: Conversion to Thermal Conductivity of Air

Table **4.1.3** lists some results for the steady-state values of the heat flux, with the unscaled and scaled heat fluxes. The simulated results were too close to allow the computation of the order of spatial convergence for the heat flux, however this will be done in the next problem.

Table 4.1.3 Planar Couette Grid Convergence of Steady State Heat Flux: 2^{nd} Order Discretization of g

Height *(m)*	**# Rows**	ΔT *(K)*	**q, simulated** W/m^2	**q, exact,** κ_{model} W/m^2	$\mathbf{q_{sim}}$**, rescaled** κ_{air} W/m^2	**q, exact** κ_{air} W/m^2	**% error**
0.008	5	3	-2.88	-2.877	-9.65	-9.6455	0.0879%
0.008			-2.88	-2.877	-9.65	-9.6455	0.0879%
0.004			-5.75	-5.755	-19.27	-19.291	0.0859%
0.002	9	3	-11.51	-11.509	-38.58	-38.582	0.0010%
0.001			-23.02	-23.0198	-77.16	-77.164	0.0010%
0.0005			-46.04	-46.0395	-154.33	-154.327	0.0010%
0.008			-2.88	-2.877	-9.65	-9.6455	0.0879%
0.004			-5.75	-5.755	-19.27	-19.291	0.0859%
0.002	17	3	-11.51	-11.509	-38.58	-38.582	0.0010%
0.001			-23.02	-23.0198	-77.16	-77.164	0.0010%
0.0005			-46.04	-46.0395	-154.33	-154.327	0.0010%
0.008			-2.88	-2.877	-9.65	-9.6455	0.0879%
0.004			-5.75	-5.755	-19.27	-19.291	0.0859%
0.002	33	3	-11.51	-11.509	-38.58	-38.582	0.0010%
0.001			-23.02	-23.0198	-77.16	-77.164	0.0010%
0.0005			-46.04	-46.0395	-154.33	-154.327	0.0010%
0.008			-0.959	-0.9592	-3.21	-3.22	0.0164%
0.004	17	1	-1.92	-1.918	-6.44	-6.43	0.0879%
0.002			-3.84	-3.837	-12.87	-12.86	0.0879%
0.001			-7.67	-7.673	-25.71	-25.72	0.0424%
0.008			-4.8	-4.7958	-16.09	-16.08	0.0879%
0.004	17	5	-9.59	-9.592	-32.15	-32.151	0.0164%
0.002			-19.18	-19.183	-64.29	-64.30	0.0164%
0.001			-38.37	-38.366	-128.62	-128.61	0.0097%

4.1.2 *Rayleigh-Bénard Convection*

Thermally, planar Couette flow is simply a conduction problem. To examine the model's convective aspect, Rayleigh-Bénard convection, a problem geometrically sim-

ilar to the planar Couette case but different in the boundary conditions, was simulated. As before, two infinite parallel plates bound a fluid. However both plates are stationary, and in contrast to the stable temperature field that occurs in planar Couette flow, here the bottom plate is at a higher temperature than the the upper plate, and any perturbation to the experiment will lead to hot fluid rising to the cold upper surface where it cools and then flows back down to the hot surface.

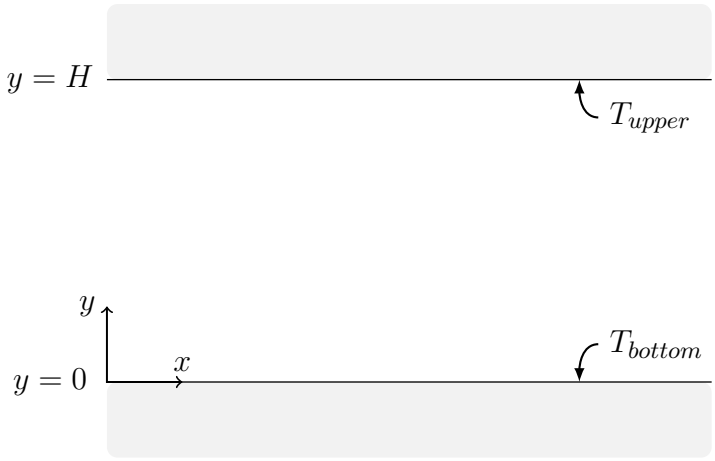


Figure 4.1.10 The Rayleigh-Bénard Convection Problem

The problem geometry is shown in figure **4.1.10**. Mathematically the boundary conditions are expressed as

$$\begin{aligned} t < 0 : u(y,t) &= 0 \\ t \geq 0 : u(H,t) &= 0 \\ u(0,t) &= 0 \end{aligned} \tag{4.18}$$

$$\begin{aligned} t < 0 : T(y,t) &= T_{bottom} \\ t \geq 0 : T(H,t) &= T_{upper} \\ T(0,t) &= T_{bottom} \end{aligned} \tag{4.19}$$

with $T_{bottom} > T_{upper}$. The CFD domain is conceptually the same as in figure **4.1.2**, with set temperatures on the upper and bottom surfaces and periodic conditions

on the sides. Following the results of Clever and Busse (1974), from whence the benchmark data for comparison is taken, the CFD domain is taken to be twice as wide as its height. In contrast to some computational studies of this problem which rely on numerical noise to provide a perturbation to initiate convective flow, a technique which can require around 10^4 timesteps for flow to just for slow motion to start (Zhou et al., 2004), this study applied a small and transient thermal perturbation to one node at the middle of the bottom plate. For this convective flow case, the dimensionless parameter of interest is the Rayleigh number

$$Ra = \frac{|\boldsymbol{g}|\beta\Delta T H^3}{\nu\alpha} \tag{4.20}$$

with $\Delta T = T_{bottom} - T_{upper}$. Use of the convergence (to steady-state) criteria given in the section on planar Couette flow led to excessively long simulation times in this situation, apparently due to memory limitations associated with Dymola. Thus for these simulations, and all simulations in this work other than planar Couette cases, the solution was considered to be steady-state if the Nusselt number, described below, was constant to 5 decimal places for several seconds of simulated time. Figures **4.1.11** and **4.1.12** depict contours of constant streamfunction – streamlines – and isotherms for three Rayleigh numbers. These plots compare qualitatively well to similar results in the literature.

In examining convective heat transfer, the Nusselt number $Nu = q/q_{conduction}$ is an important figure of interest: being the ratio of actual heat transfer across the fluid domain to the heat transfer that would occur if conduction across the fluid domain were the only available transfer mechanism, Nu gauges the 'strength' of convective heat transfer and serves as a nondimensional measure of the heat transfer itself. Table **4.1.4** gives steady-state Nusselt numbers at the upper and bottom plates for various

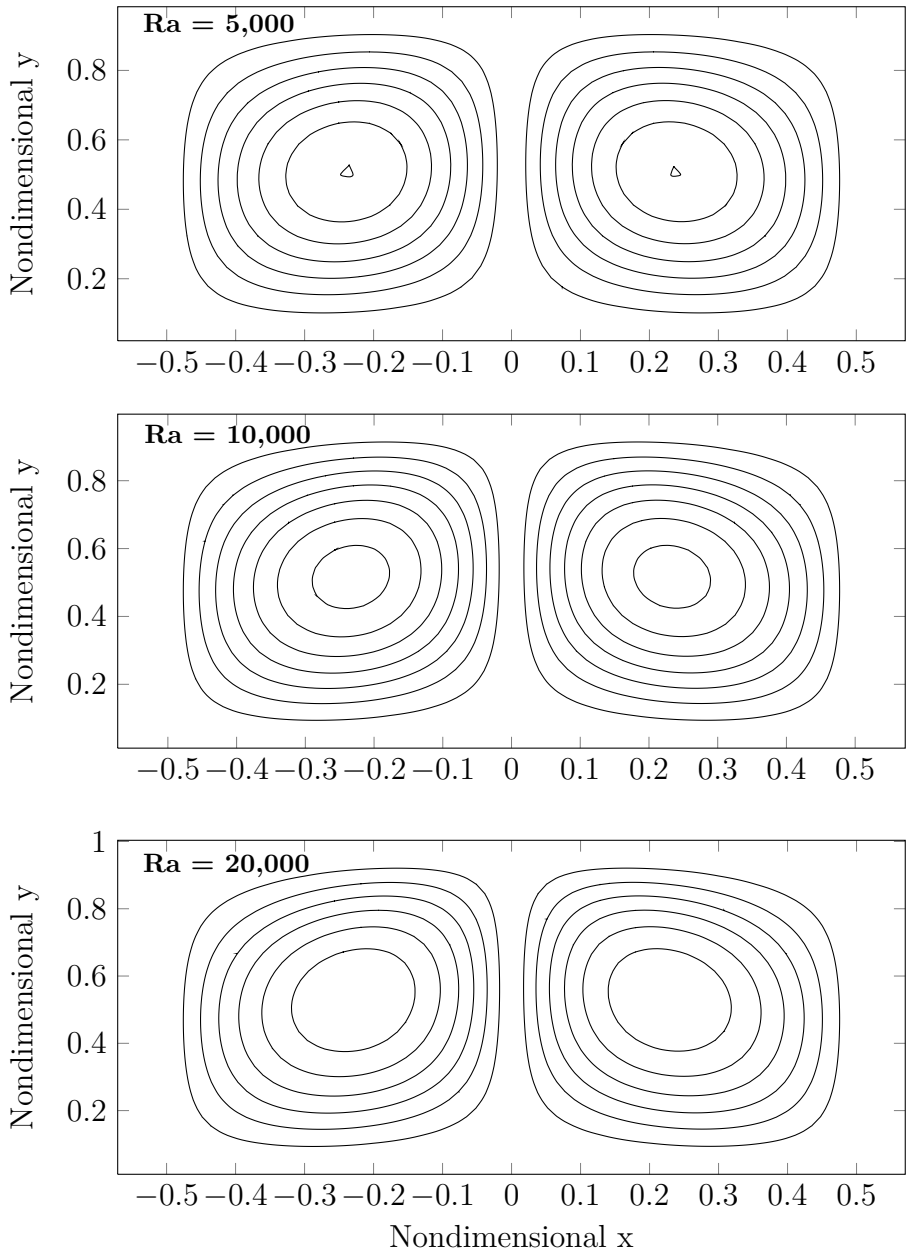


Figure 4.1.11 Rayleigh-Bénard Streamlines

configurations and Rayleigh numbers and compares them to benchmark data from Clever and Busse (1974).

During this work it was found that grids whose total number of nodes was above approximately 2700 would fail to simulate in Dymola, with the practical consequence that a grid convergence study could not be carried out to the error levels reported for

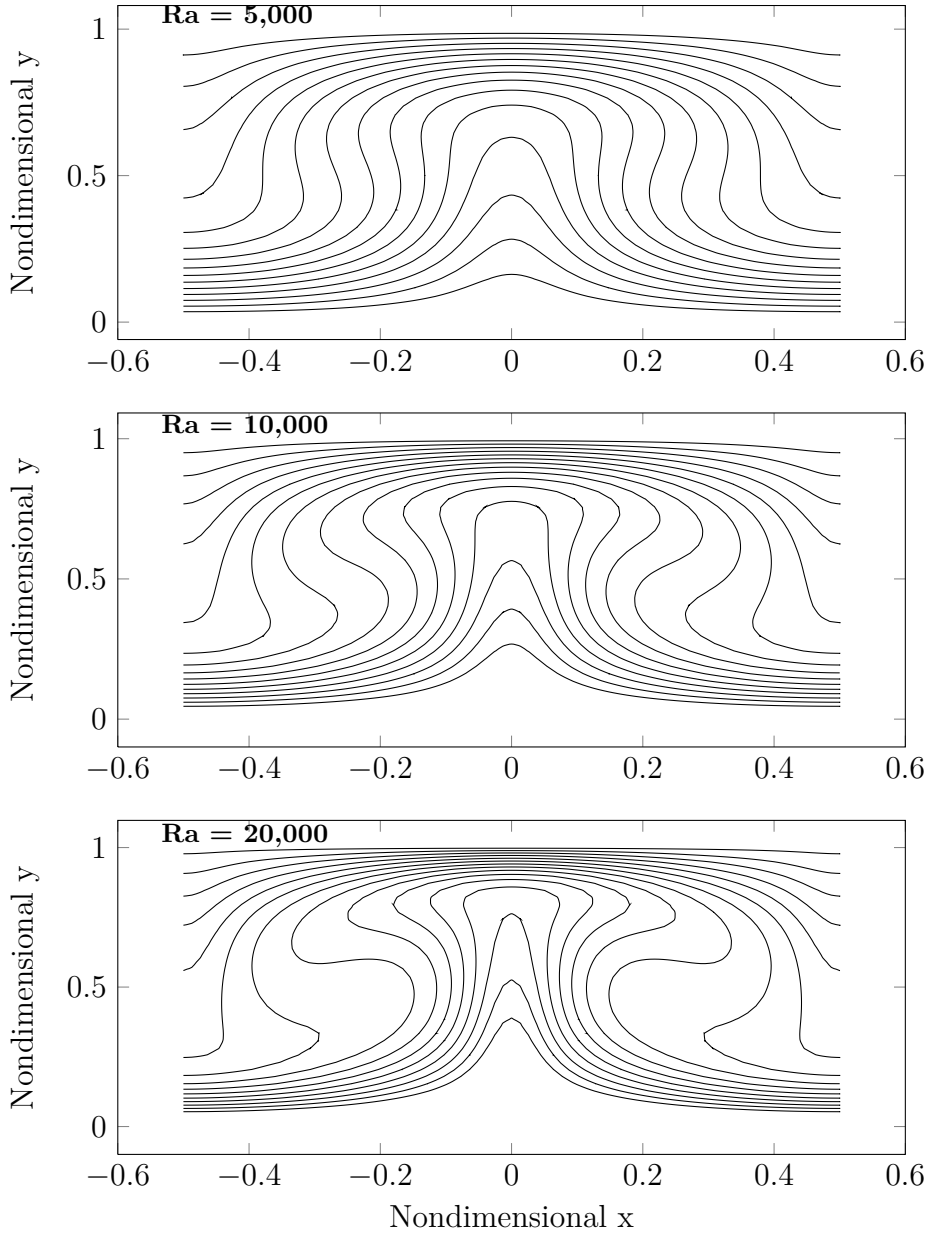


Figure 4.1.12 Rayleigh-Bénard Isotherms

planar Couette flow. Additionally the grid was not doubled as in the planar Couette case and the grid ratio used for determining the order of grid convergence Υ was not constant. For this case Υ can still be calculated by

$$\Upsilon = \omega(\Upsilon_{prev}) + (1-\omega)\left\{\frac{\ln\left[\frac{(n_{12}^{\Upsilon_{prev}}-1)(Nu_3-Nu_2)}{(n_{23}^{\Upsilon_{prev}}-1)(Nu_2-Nu_1)}\right]}{\ln(n_{12})}\right\} \tag{4.21}$$

where $\omega \sim 0.5$ is a relaxation parameter and the numerical subscripts indicate the

Table 4.1.4 Bénard Convection: Variations in Rayleigh Number

Height *(m)*	**# Rows**	**Ra**	**Δ T** *(K)*	**Nu_u**	**Nu_b**	**Nu_{bnchmk}**	**% error**
0.02	35	5000	6.1	2.034	2.034	2.116	-3.878%
0.0125	35	5000	25	2.081	2.081	2.116	-1.64%
0.02	35	10000	12.2	2.518	2.518	2.661	-5.38%
0.013	38	10000	44.4	2.609	2.609	2.661	-1.95%
0.03	35	20000	7.23	2.655	2.655	3.258	-18.25%
0.016	40	20000	47.7	3.135	3.135	3.258	-3.78%

coarseness/fineness of grids, with lower numbers representing finer grids. Obviously equation **4.21** must be iterated to come to a solution (Roache, 1998). Results of this iteration are given in table **4.1.4**, which shows that the order of grid convergence for the Nusselt number is 1.89.

Table 4.1.5 Bénard Convection: Grid Convergence for Ra = 5000

Grid	**H** *(m)*	**Nu (this work)**	**Nu (benchmark)**	**% error**	**ϒ**
21 × 41		1.899		-10.26%	
31 × 61	0.02	2.013	2.116	-4.87%	1.89
35 × 69		2.034		-3.88%	

One inference that can be drawn from the results in table **4.1.4** is that the grid must become finer as the Rayleigh number increases. With the limitations imposed by Dymola, this resulted in the inability to explore larger Rayleigh numbers.

4.2 Test Cases: Conjugate Models and Tangelo Coupling

Whereas the two situations considered above were treated as purely fluid mechanical

problems simulated by CFD and provide some measure of verification of the lattice Boltzmann method as implemented here, the following problems demonstrate semi-conjugate or 'tangelo' coupling as well as exemplify some simple multi-(discipline or subject) domain capabilities.

4.2.1 *Thermal Planar Couette Flow, Modeled as a Laboratory Experiment*

On its own, CFD is typically only concerned with what happens inside a fluid domain and is unconcerned with anything outside this realm: the only connection to the larger universe is the boundary conditions. For example, bounding no slip/no penetration surfaces are conceptually viewed as infinitesimally thin sheets which prevents mass flow, have a specified temperature or heat flux, and otherwise have no computational life. This is the attitude taken in the first two cases above, and for many problems of interest this is an appropriate choice. However for investigations of systems composed of other interacting subsystems, which include fluid subsystems, the system boundaries must be expanded beyond the traditional CFD realm.

In this subsection the planar Couette case is simulated with these boundaries expanded to represent this situation as might be implemented as a physical laboratory experiment, albeit in a simple way. Consider an apparatus with a stationary wall and a moving wall, both of which are of finite thickness and have material properties such as thermal conductivity, specific heat, and density. In a physical experiment, the infinite extent of the plates might be approximated by using a circular Couette apparatus – two concentric cylinders with fluid being sheared in the space between the cylinders – whose average radius is much larger than the gap between the cylinders. For the Couette flow, the temperatures of the surfaces in contact with the sheared fluid are to be set to given values. Consider however that the experimental apparatus heats or cools the plates from the outside surfaces by means of a water bath, i.e. the surfaces not in contact with the sheared fluid are in contact with a circulating thermal control

fluid. Although one could fairly easily determine the required temperatures of the outside surfaces as well as the temperature of the bath if a convective heat transfer coefficient h is known, consider for illustrative purposes that the outside surfaces' temperatures are to be actively monitored and controlled in an active feedback loop by controlling the temperature of the baths. Conceptually this situation is represented in figure **4.2.1**.

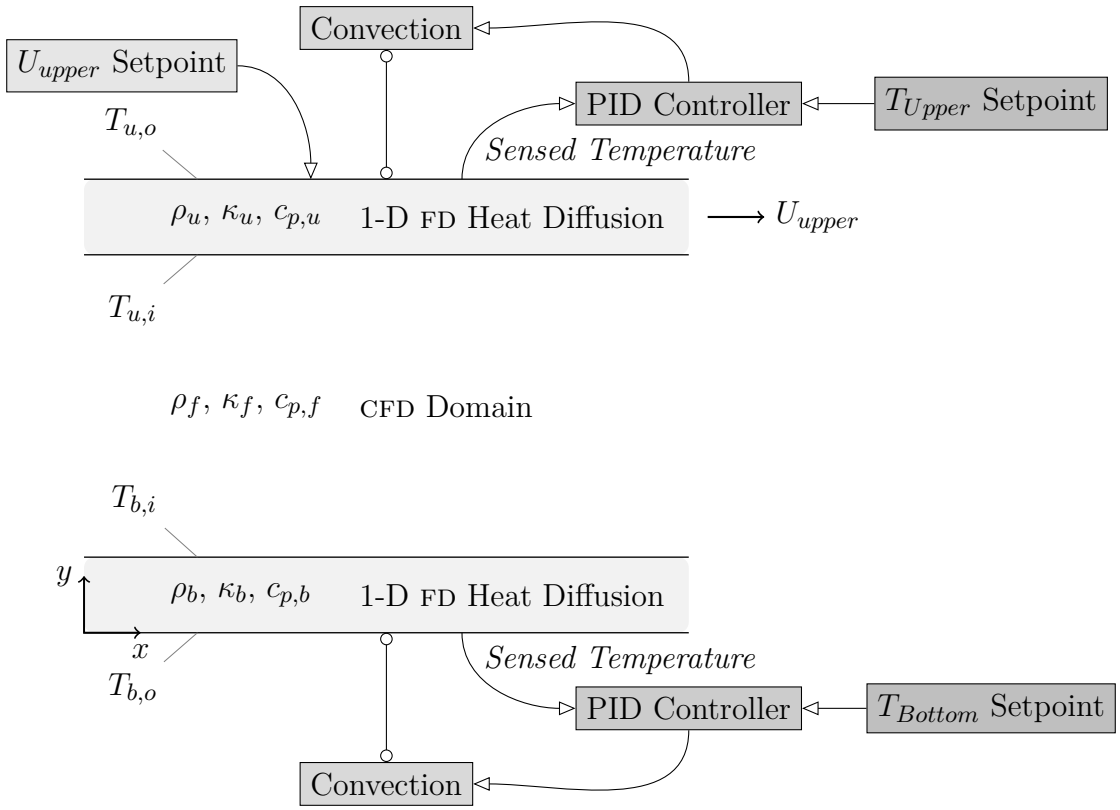


Figure 4.2.1 The Conjugated Planar Couette Problem

Figure **4.2.1** is also, not incidentally, a representation of the Modelica model for this situation. For the water baths, convection models were created representing Newton's law of convective heat transfer; models of the walls were created using a one-dimensional finite difference scheme for the (conductive) heat diffusion equation. These models are given in Appendix B. The proportional-integral-derivative (PID)

controller used was from the Modelica Standard Library, and the setpoint models simply fed a number into the models they are connected to.

The links in figure **4.2.1** indicate certain modeling protocols: the open triangular arrows indicate connections with a directionality, i.e. using `input` or `output` Modelica keywords; the open circles terminating the other connectors indicate acausal connectors linking temperatures and heat fluxes, the latter being a `flow` variable in this case involving a isothermal surfaces of infinite extent.

A general unsteady solution is unknown due to the presence of the PID controller, however in steady state the problem is once again simply one of conduction. Because $T_{u,o}$ and $T_{b,o}$ are known, the unknown temperatures $T_{u,i}$ and $T_{b,i}$ are easily found to be

$$\begin{aligned} T_{u,i} &= \frac{\Delta y_u}{\kappa_u}\frac{T_{b,o} - T_{u,o}}{R_{val}^{tot}} + T_{u,o} \\ T_{b,i} &= T_{b,o} - \frac{\Delta y_b}{\kappa_b}\frac{T_{b,o} - T_{u,o}}{R_{val}^{tot}} \end{aligned} \tag{4.22}$$

where Δy_u and Δy_b are the thicknesses of the upper and lower plates, respectively, and R_{val}^{tot} is the total R-value of the two plate and fluid combination:

$$R_{val}^{tot} = \sum_i \frac{\Delta y_i}{\kappa_i} \tag{4.23}$$

with $i = \{u, f, b\}$. Being steady-state, the temperature profile within each material is linear. The convection models, being expressions of Newton's law of cooling, leads to

$$\begin{aligned} T_{u,amb} &= \frac{1}{h}\frac{T_{b,o} - T_{u,o}}{R_{val}^{tot}} + T_{u,o} \\ T_{b,amb} &= -\left(\frac{1}{h}\frac{T_{b,o} - T_{u,o}}{R_{val}^{tot}} - T_{b,o}\right) \end{aligned} \tag{4.24}$$

Material parameters used are given in table **4.2.1**. For both of the the convection models, the convection coefficient h was taken to be 4 W/(m^2 K). Taking $T_{u,o}$ and

$T_{b,o}$ to be 298.15 K and 293.15 K, respectively, the results of the exact solution in from equations **4.22**, **4.23**, and **4.24** as well as a conjugated simulation are given in figure **4.2.2** and table **4.2.2**.

Table 4.2.1 Parameters for the Conjugate Planar Couette Example

Part	ρ (kg/m^3)	κ $W/(m\ K)$	c_p $J/(kg\ K)$
Upper Plate	100	0.05	100
Fluid	1.20	0.0074	287
Bottom Plate	50.0	0.025	50

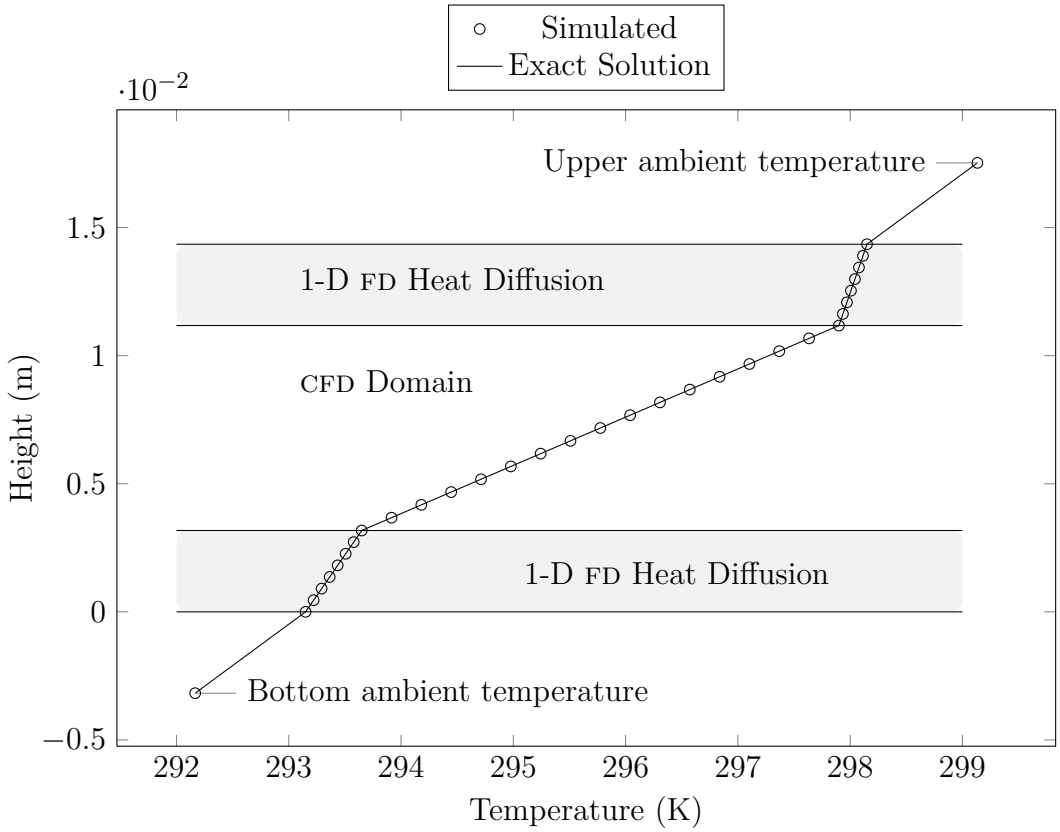


Figure 4.2.2 The Conjugated Planar Couette Problem: Temperature Profile

Table 4.2.2 Conjugated Planar Couette Heat Flows

Location	**Heat Flux** *(W/m^2)*		**% Error**	**Heat Flux/δx** *(W/m)*		**% Error**
	Exact	*Simulated*		*Exact*	*Simulated*	
$T_{u,amb}$		–	–		0.007869856	-0.00003%
$T_{u,i}$	3.934927	3.934923	0.00009%	0.007869853	0.007869856	-0.00003%
$T_{b,i}$		3.934924	0.00005%		0.007869849	0.00005%
$T_{b,amb}$		–	–		0.007869855	0.00002%

This case demonstrates the expansion of the boundaries of a CFD simulation through the combination of a CFD model with conductive heat transfer as well as a simple control scheme.

4.2.2 *Convection in a Square Cavity with Set Temperatures on Side Walls and Adiabatic Top and Bottom Walls*

The previous problems used CFD domains that were of infinite extent in one direction. To examine solutions using a more realistic geometry, the problem of natural convection in a square cavity is simulated. Here the two vertical surfaces of a cavity have set temperatures while the horizontal floor and ceiling surfaces are adiabatic as shown in figure **4.2.3**. Here the left vertical wall is the hot surface while the right vertical surface is the cold one.

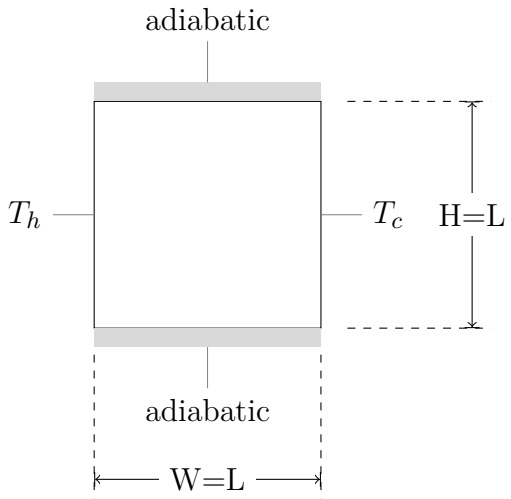


Figure 4.2.3 The Cavity Convection Problem

In the interest of exploring multi-domain modeling which includes CFD, however, this problem is modeled in a semi-conjugate or 'tangelo' fashion by connecting each node on the horizontal surfaces to a conduction model and a convection model. These features approximate the use of insulation on these surfaces and also allow the horizontal surfaces to be non-isothermal. Figure **4.2.4** visually depicts the Modelica model of this problem.

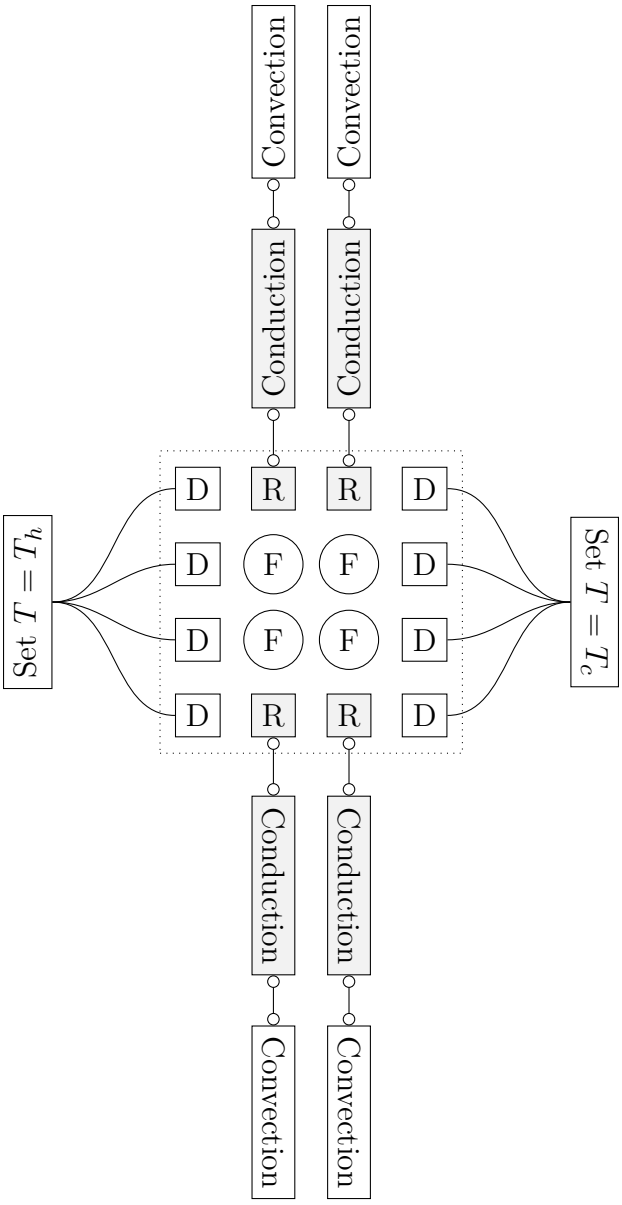


Figure 4.2.4 The Cavity Convection Problem in a Semi-Conjugate Fashion: Modelica Scheme

The results to be given were computed using a grid size of 52 × 52 lattice nodes. Figure **4.2.5** depicts streamlines and isotherms for a Rayleigh number of 10,000, and Figure **4.2.6** gives streamlines and istotherms for Ra = 100,000. Both patterns are in qualitative agreement with corresponding figures published in the literature, for example that of the benchmark data of Davis (1983). Most of the isotherm lines are normal to the floor and ceiling surfaces, in line with adiabatic conditions, although

some isotherms are not: this model does not use true adiabatic conditions, rather it only models an adiabatic surface as it might be approximated in a laboratory experiment.

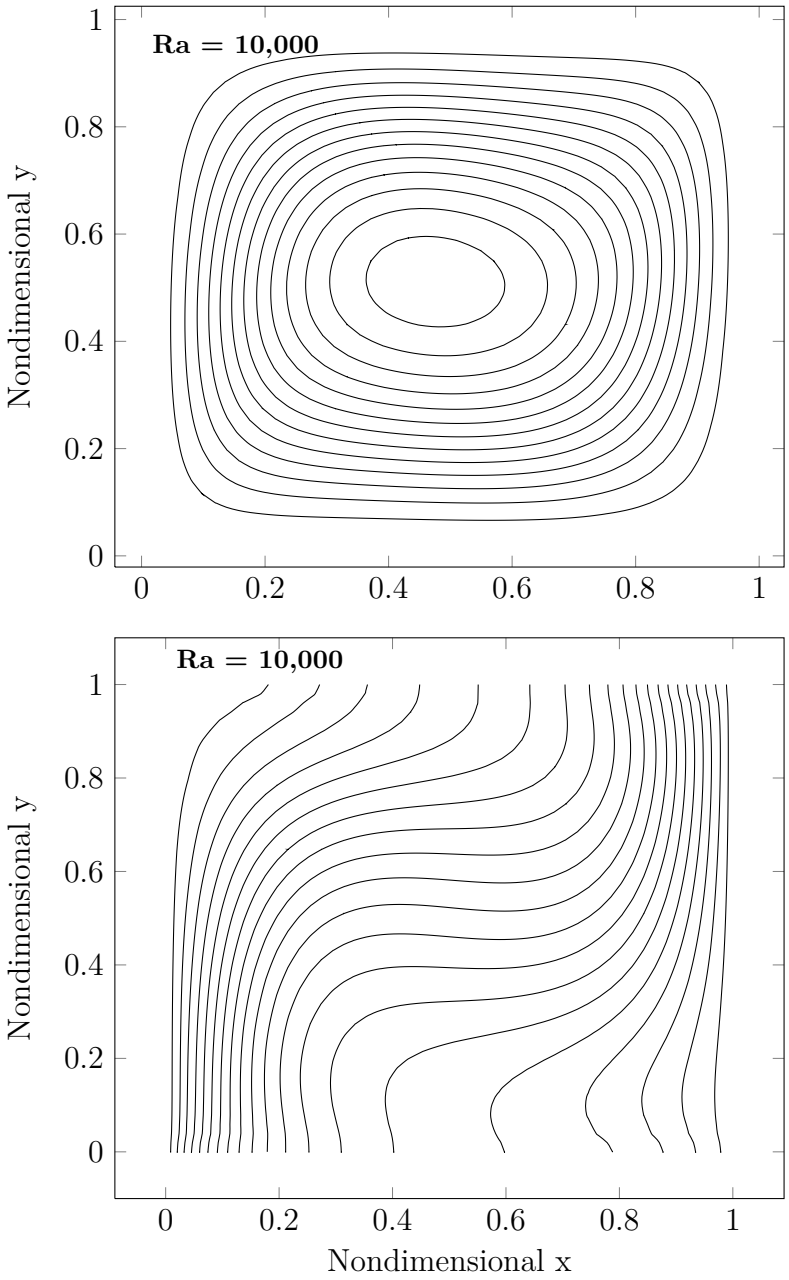


Figure 4.2.5 Cavity Convection, Ra = 10^4

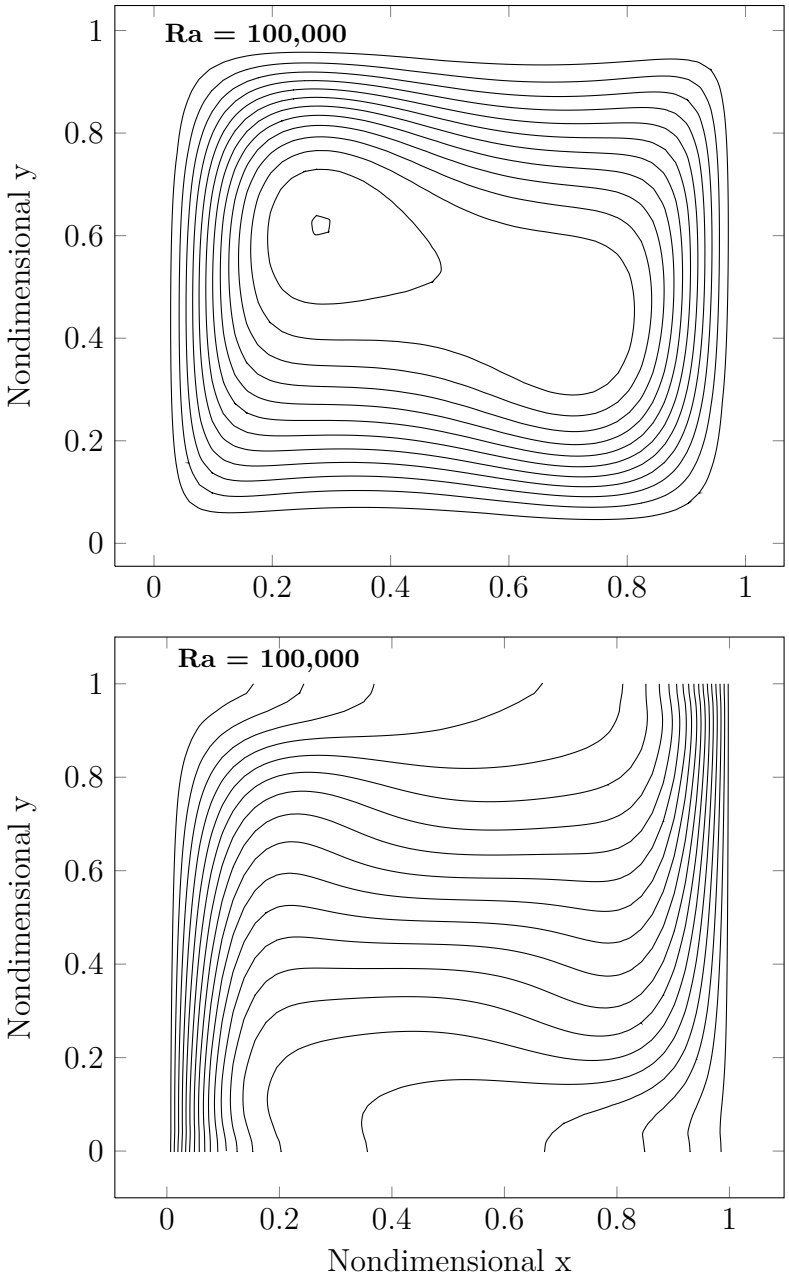


Figure 4.2.6 Cavity Convection, Ra = 10^5

More quantitative comparisons are given in tables **4.2.3** (for Nusselt numbers) and **4.2.4** (for velocities/streamfunction). For the latter table, the data are scaled following Davis (1983): the length scale is taken to be L, the scale for streamfunction ψ is taken to be the thermal diffusivity α, and time is scaled with $\frac{L^2}{\alpha}$. The Nusselt numbers $\mathrm{Nu_l}$ and $\mathrm{Nu_r}$ are the average Nusselt numbers for the left and right sides, respectively; $\mathrm{Nu_{max}}$ and $\mathrm{Nu_{min}}$ are the maximum and minimum Nusselt numbers on

the left vertical wall. The velocity u_{max} is the maximum horizontal velocity on the vertical midplane of the cavity, and v_{max} is the maximum vertical velocity on the horizontal midplane. $|\psi|_{mid}$ is the scaled value of the streamfunction at the center of the cavity, and $|\psi|_{max}$ is the maximum value of streamfunction in the CFD domain.

Benchmark results are from (Davis, 1983)

Table 4.2.3 Nusselt Numbers for Cavity Convection Case

	Ra = 10^4			**Ra = 10^5**		
	Present Work	**Benchmark**	**% Error**	**Present Work**	**Benchmark**	**% Error**
Nu_l	2.194	2.238	-1.97%	4.385	4.509	-2.75%
Nu_r	2.193	–	–	4.385	–	–
Nu_{max}	3.957	3.528	12.16%	7.405	7.717	-4.04%
Nu_{min}	-0.054	0.586	-109%	0.0172	0.729	-97.6%

Table 4.2.4 Key Scaled Velocities and Streamfunction Values for Cavity Convection Case

	Ra = 10^4			**Ra = 10^5**		
	Present Work	**Benchmark**	**% Error**	**Present Work**	**Benchmark**	**% Error**
u_{max}	15.645	16.178	-3.29%	35.64	34.73	2.62%
v_{max}	19.649	19.617	0.16%	68.84	68.59	0.36%
$\|\psi\|_{mid}$	4.591	5.071	-9.47%	8.932	9.111	-1.96%
$\|\psi\|_{max}$	–	–	–	9.726	9.612	1.19%

Most of the results show % errors less than 4%, acceptable for engineering accuracy. However some of the point measurements of the Nusselt number show large errors. These values occurred near the corners, although not at the corners. Their effect seems to be minimal on the overall results however, yet these discrepancies remain a topic for investigation.

4.2.3 *Natural Convection in a Cavity: A Single Conjugated Wall*

The final case to be considered is the problem of natural convection in a square cavity with a single conjugated wall, a configuration studied by Kaminski and Prakash

(1986). This situation is similar to the previous one with the exception that the hot left wall is now considered to have a finite thickness t with properties density ρ_w, thermal conductivity κ_w, and specific heat $c_{p,w}$ as shown in figure **4.2.7**.

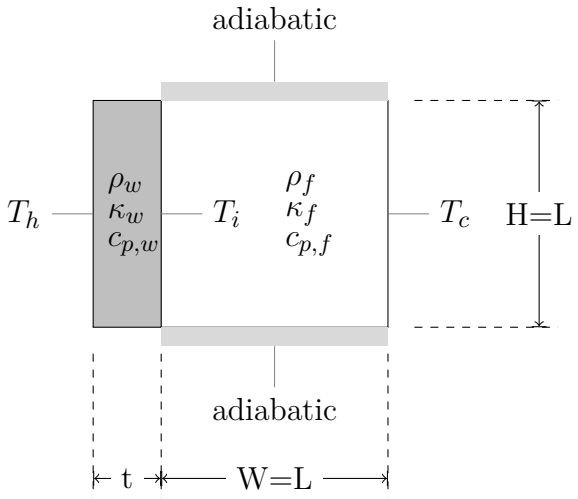


Figure 4.2.7 The "Kaminski" Problem

The approach to this problem taken here is to model it like a laboratory apparatus, as in the previous two cases. In particular the finite thickness wall is heated as was done in the planar Couette as a laboratory experiment problem, only now the inner rather than the outer wall temperature is sensed by the PID controller, as shown in figure **4.2.8**.

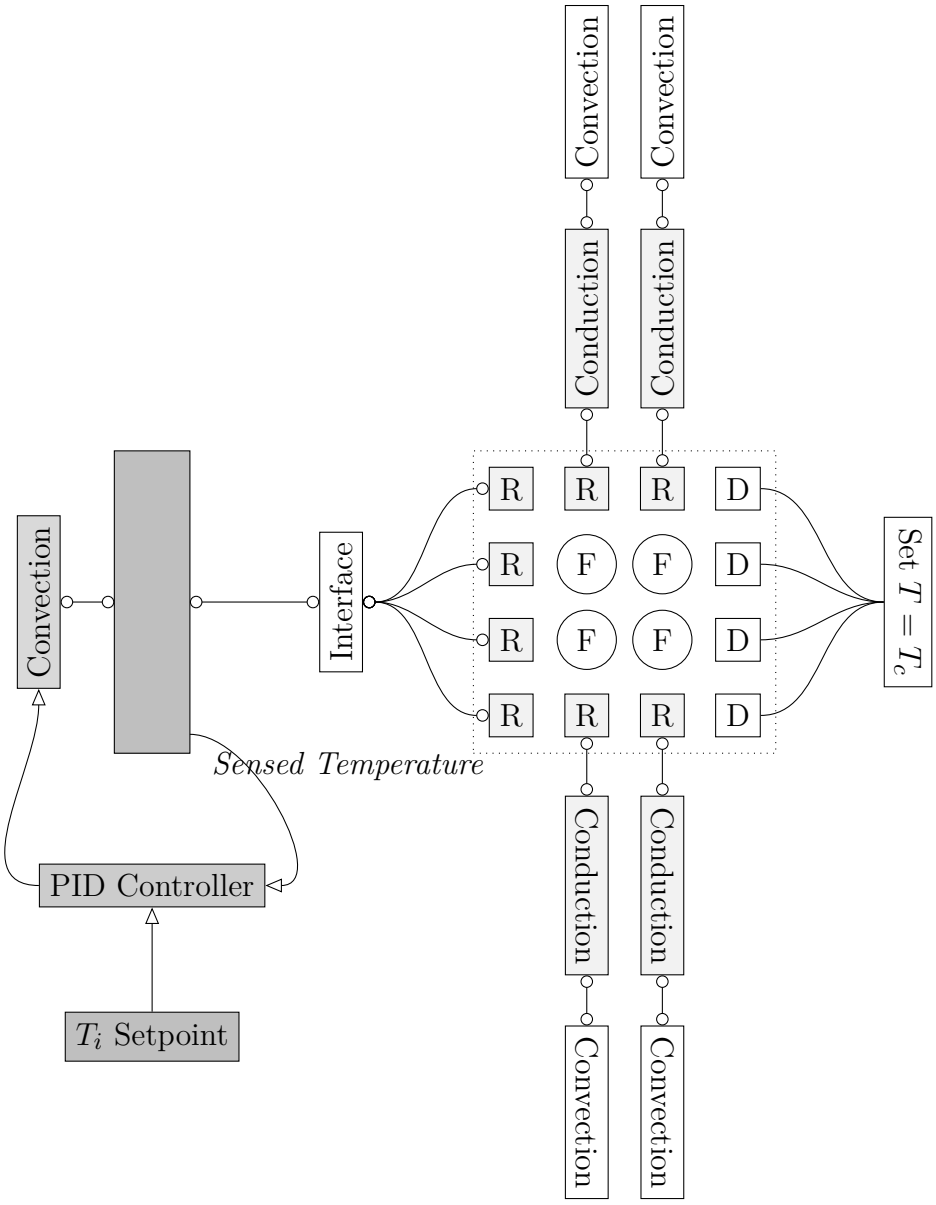


Figure 4.2.8 The "Kaminski" Problem: Modelica Scheme

However, it was found that using a grid size of 52×52 was beyond what Dymola could simulate in this case. In order to simulate this problem at the highest possible grid size, the Robin boundary nodes on the top and bottom were replaced with adiabatic nodes, eliminating the conduction and convection elements approximating laboratory insulation, as shown in figure **4.2.9**.

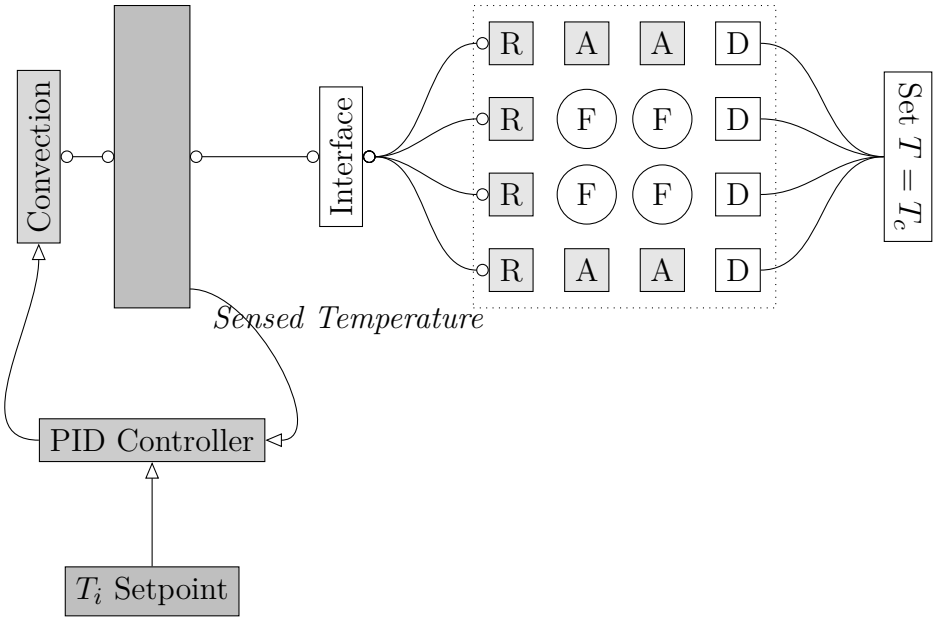


Figure 4.2.9 The "Kaminski" Problem: Simplified Modelica Scheme

For this problem the results are presented in terms of the Grashof number

$$Gr = \frac{|\boldsymbol{g}|\beta\Delta T L^3}{\nu^2} \tag{4.25}$$

where $\Delta T = T_i - T_c$, in conjunction with the ratio $\frac{\kappa_w L}{\kappa_f t}$. Table **4.2.5** gives values of the overall Nusselt numbers for two configurations, and figures **4.2.10** and **4.2.11** give streamlines and isotherms for these cases. Results compare favorably to those of Kaminski and Prakash (1986).

Table 4.2.5 Overall Nusselt Numbers for the Kaminski Problem

Gr	$\frac{\kappa_w L}{\kappa_f t}$	$\mathrm{Nu}_{\mathrm{sim}}$	$\mathrm{Nu}_{\mathrm{benchmark}}$	% Error
10^3	5	0.88	0.84	4.76%
10^5	50	3.71	3.67	1.09%

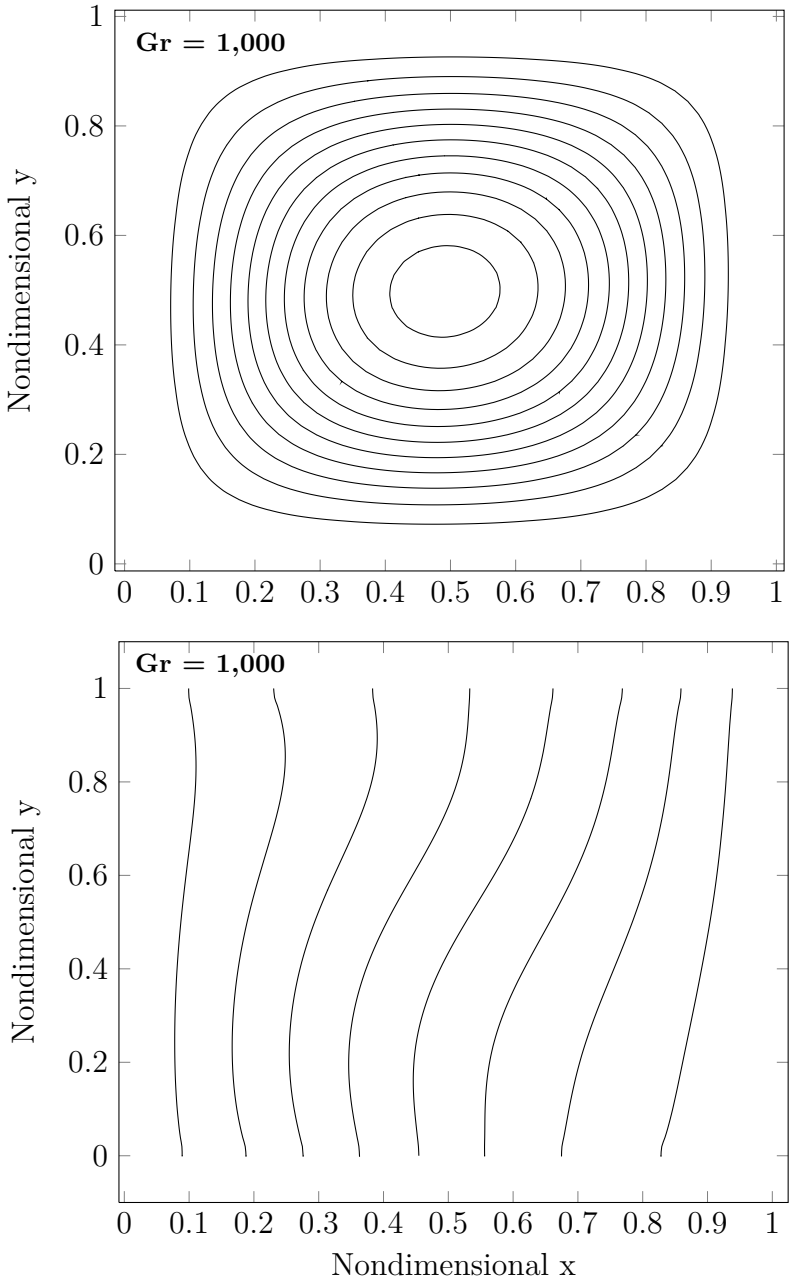


Figure 4.2.10 Kaminski Problem, Gr = 10^3, $\frac{\kappa_w L}{\kappa_f t}$=5

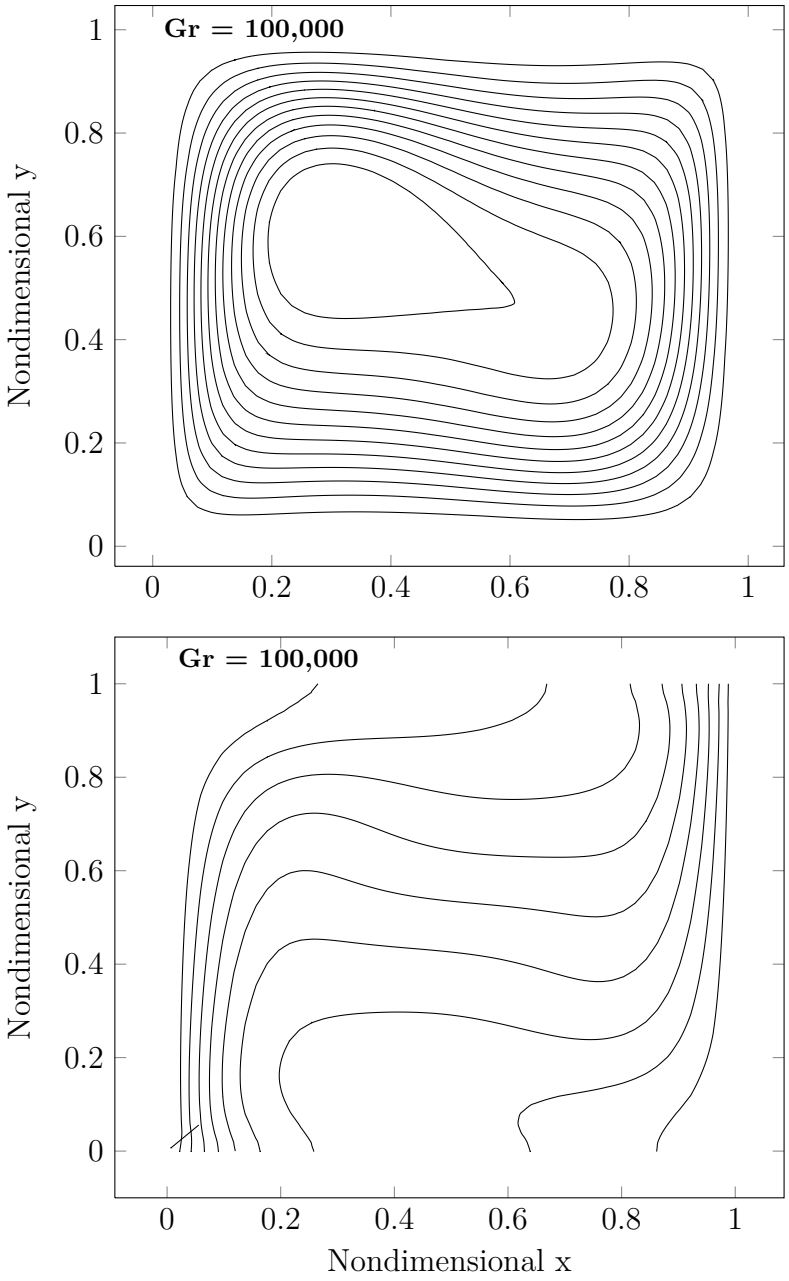


Figure 4.2.11 Kaminski Problem, Gr = 10^5, $\frac{\kappa_w L}{\kappa_f t}$=50

4.3 Summary

In this chapter the Modelica implementation of a lattice Boltzmann CFD scheme has been used to simulate several test cases which have been compared to analytic solutions and benchmark data, both for pure CFD simulations and CFD simulations coupled to conductive heat transfer and simple controls. Comparisons to planar Couette

flow solutions show verification; considering benchmark data to be experiments, the conjugated solutions show some measure of validation, although software limitations prevented fuller explorations.

CHAPTER 5: DISCUSSION

5.1 Contribution

Equation based modeling languages such as Modelica have been developed i) to enable the *modeling* of physical systems without the labor of *programming* those models in an algorithmic language, and ii) to enable models from separate domains to be modeled together with their interactions as they exist in real systems. Buildings, being complex assemblages of many different interacting subsystems spanning multiple domains, represent one type of system that can benefit from such an approach. Of particular interest in this context is the intersection of air flows, heat transfer, and controls, domains which in the past have been treated either separately or with only cursory attention. In this work it has been demonstrated that one type of airflow modeling method, CFD, can be integrated with models of heat transfer and controls within the context of equation based modeling. Although Modelica models are typically of the lumped parameter and one-dimensional type, and although previous work has explored the solution of partial differential equations enabling multidimensional problems to be solved (Saldamli et al., 2005), the work presented in this thesis represents the first time to the author's knowledge that such models, and in particular CFD models, have been implemented in a pure Modelica framework with the inclusion of models from different domains.

In particular, while the coupling of CFD to heat transfer processes (through ES) remains a topic of research, many of the approaches followed thus far are all of a similar type: internal/external coupling paradigm, the use of coupling variables which are intermediate to the actual physical quantities that are manifest in the 'real' coupling, quasi-dynamic/ping-pong/loose coupling vs. full dynamic/onion/tight coupling. This work represents a new way in which the paradigm is neither internal nor external

but offers the benefits of both, the coupling variables more directly represent the physical mechanisms of the coupling, and the coupling strategy, while tight in its effect, requires no explicit coding of iteration between separate programs to achieve consistency between models.

External coupling is touted as leading to a more maintainable (federated) simulation infrastructure than internal coupling since each program in the coupled federation of programs is allowed to evolve on its own; the only thing to be maintained by those wishing to couple programs together is the interface between the programs (Djunaedy et al., 2005). This is a worthwhile goal, however in the context of a hierarchical modeling language such as Modelica, the semantic distinction between external and internal coupling applies less than in the context of hard-wired programs. Given that the creation of interfaces between models and the organization of models into packages are fundamental features of the language, it is reasonable to expect that models and packages of models can likewise evolve independently yet still come together in a greater model so long as the interfaces are similarly maintained if necessary, in effect as the federated programs do in external coupling. This is indeed what happened during the evolution of the lattice Boltzmann models in this work: the model of heat conduction in solids was never altered – only new models with sensor ports were extended from more basic models – yet the lattice Boltzmann models underwent many different revisions independently of other models, in particular with respect to boundary conditions/edge nodes and the scheme for the evolution of the internal energy distributions $g_i \rightarrow gm_i$, yet these changes had no effect on the infrastructure needed to combine the models. In this practical respect the paradigm might be considered external, however in a formal respect the coupling is internal in the sense that all models are expressed in a common language of physical system description.

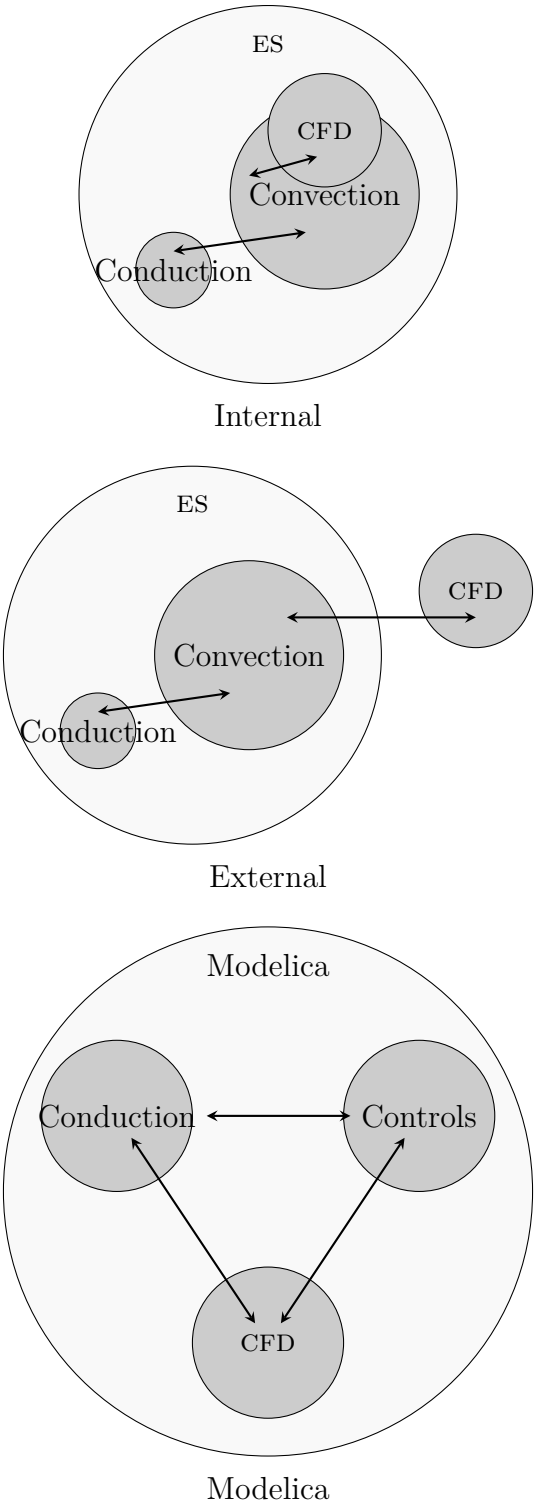


Figure 5.1.1 Three Coupling Paradigms

The ability to combine models is one of the key reasons for the existence of modeling languages like Modelica, and this ability arises due to the features for creating links between models at the language level. Previous work in coupling CFD to ES has dealt with at least one pre-existing simulation program, e.g. EnergyPlus or ESP-r, with the underlying premise that no similar linking mechanism exists as a fundamental given. This, combined with the overlap between CFD and ES (in its heat transfer elements) requires that any coupling between the two programs must be knowledgeable of and consistent with the internal representation of heat transfer in each program. In the current case, given the boundary condition strictures of traditional CFD, this requires that either the coupling be done using the intermediate variable h which is determined by CFD, or by iterating temperatures T and Q between CFD and ES. In this traditional coupling it is considered preferable to use the intermediate varible h, although it is possible for h to have a negative value in certain flow cases which can cause the direction of heat flow to be incorrect (Zhai, 2003). Modelica's connector construct together with the flow (and stream) prefixes enables the creation of links between models which are acausal and represent the actual physics without needing to be concerned with how the physics are represented in the constituent models.

From a modeler's perspective, the benefits of such an equation based approach are most clearly seen in the connection between CFD and wall conduction. All that is required is that each model describes the temperatures and heat fluxes/heat flow rates at their boundaries. In fact, the conversion from a Dirichlet edge node in the CFD package to a Robin-type node only required changes to a few lines to define the heat flux/heat flow rate appropriately and link that equation to an appropriate connector in the model. The coupling variables are now the physical ones of interest, and no explicit coding is required for the linked models to have a consistent solution as depicted in figure **5.1.2**.

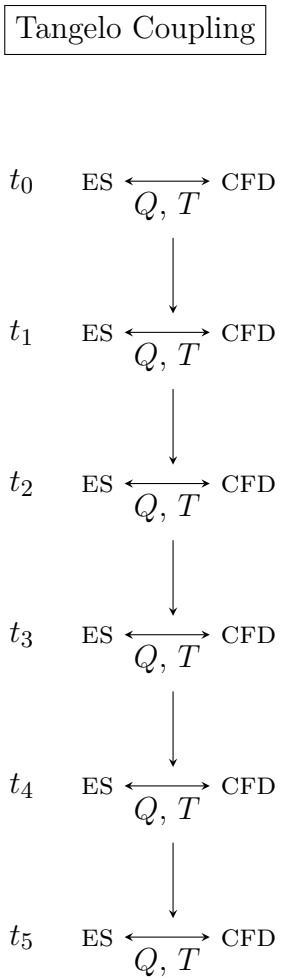


Figure 5.1.2 Tangelo Coupling

The lattice Boltzmann approach to CFD used here is well-suited as a fluid-mechanical description in the context of Modelica. It describes the time-evolution of flows which is in line with Modelica's aim to enable descriptions of physical dynamics. Furthermore it does so in a relatively simple fashion, without iteration, yet nevertheless is capable of providing a large amount of information: in particular the heat flux is a natural product of the model, although a true second order scheme is required for the internal energy distributions g_i, and is not a quantity derived from another macroscopic variable. In fact it is curious that this aspect of the lattice Boltzmann method has been used as little as it has – for example Guo et al. (2002), Guo et al. (2007), and Li et al. (2008), among others, resort to the calculation of the derivative of the

temperature field for the specification of heat flux (typically only for adiabatic cases) at a boundary in traditional Neumann style. In addition to this work, only D'Orazio et al. (2004) seems to have used heat flux as a moment of a distribution function, $\boldsymbol{q} = \int g d\boldsymbol{\xi}$ as the basis of a boundary condition. In addition, the possibility of improved turbulence models is a potential benefit, although such a hope is conjectural at this point.

All of these features combine to yield a framework for the combination of many types of models beyond CFD and heat transfer; the simple examples given in this thesis also incorporate (simple) PID controls and mimic benchmark fluid mechanics problems not as abstract, tightly delineated problems divorced from the rest of the universe as they would be treated in commercial CFD programs, but as how those problems might actually be run in a laboratory experiment.

5.2 Limitations and Directions for Future Work

Although this work represents a new way of coupling CFD with other modes of heat transfer, and with other domains as well, it is nevertheless quite limited as a practical tool. First, any applications are limited to laminar flows and to cases where the flow region does not require the number of nodes to go beyond Dymola's – or any other tool which implements Modelica models – apparent capabilities, as was seen in Chapter 4. Furthermore, the current approach is a discrete time integration with a fixed step size which links the grid spacing and the timestep through the lattice velocity $\check{\xi}$. Increasing the grid count/decreasing grid spacing as may be needed for higher Rayleigh/Grashof number flows, simultaneously decreases the timestep size, which in application can lead to excessively long simulation times. In addition, the timescales associated with these different processes can differ by orders of magnitude (Zhai, 2003) which can lead to stiffness, a common problem in the conflated simulation of heat and airflow using DAE based methods such as in the present work (Sahlin,

2003). Some of the causes of stiffness and short timesteps are inherent to the physics being described or simulated, but short timesteps should be avoided to the extent possible at the model creation and computational execution levels.

With respect to model creation, the lattice Boltzmann method is but one numerical scheme for solving the Boltzmann equation. The congruent discretization of physical and (molecular) velocity spaces is simply a convenient choice, one that is historically rooted in the lattice Boltzmann method's origins in cellular automata. However it is not necessary to discretize these spaces in this manner, as was described early in the realization that lattice Boltzmann can be viewed as a finite-difference scheme for the continuous Boltzmann equation. Indeed, in one of the first papers pointing this out, He and Luo (1997a) developed a finite difference Boltzmann scheme in which the discretization of physical space and velocity space were decoupled and used this to simulate flow over a backward-facing step using nonuniform grids. Since then many workers have constructed discretized Boltzmann schemes which can use nonuniform grids, thereby speeding up simulation, as well as in some cases represent curved boundaries using, for example, body-fitted coordinate systems, for example Mei and Shyy (1998) and Guo and Zhao (2003) for athermal flows, Li et al. (2008) for a double-distribution thermal model, and Surmas et al. (2009) for multispeed thermal models.

The discretization scheme is not limited to finite differences, however. Efficient finite element schemes have been proposed (Lee and Lin, 2001), as well as finite volume (Stiebler et al., 2006). Methods such as these have the advantage that the grids can conform to geometries not easily described by the mathematical mappings characteristic of body-fitted coordinate systems.

Much research nevertheless still goes into ways to make the traditional lattice Boltzmann scheme, with it's congruent physical and velocity space discretizations (leading to what are sometimes called 'space filling' lattices), more efficient. These

traditional schemes remain popular due to their relative ease of implementation and their natural affinity for parallelization (Chen and Doolen, 1998). Filippova and Hänel (1998) initiated this branch of development with their method for the treatment of curved boundaries not coincident with lattice nodes and for their multi-block method of local lattice refinement, all within the context of the original lattice Boltzmann discretization of physical and velocity spaces. This work has since been refined, for example see the review article of Yu et al. (2003).

These issues will become much more pronounced as three-dimensional models are developed, where the number of molecular velocities for the f_i and g_i distributions increases from 9 to 15 or greater.

Further approaches for the speedup of these simulations include the fast evaluation of equilibria (Chikatamarla et al., 2006), and execution of the Modelica models in a parallel computing environment such as graphics processors, perhaps taking advantage of the recent additions to the Modelica language allowing for the "mapping of models to execution environments" (The Modelica Association, 2010).

Another approach tried during the course of this thesis was the method of lines (Cellier and Kofman, 2006). This technique converts partial differential equations not into algebraic systems of equations but rather into ordinary differential equations. In the context of a time-domain simulation such as implied by Modelica, this entails converting spatial derivatives into, e.g., finite-difference approximations, while leaving the temporal derivatives to be solved by the ODE integrator, for example the default solver in Dymola, DASSL. In fact the model of wall conduction used in the semi-conjugate 'tangelo' cases of Chapter 4 can be considered to be a method of lines model of the one-dimensional heat diffusion equation (Appendix B gives this model). The conceptual advantage of this approach is that by not using a fully discrete approach, the ODE integrator can, if capable, apply variable time steps and thereby speed up the simulation times compared to models with fixed time steps such as those used in this

work. Preliminary trials simulating athermal planar Couette and lid-driven cavity flow using the method of lines, however showed severe instabilities at low Reynolds numbers. The cause of this instability was not investigated to conclusion, however a potential culprit could be that the ODE integrator used a step size so large that $\frac{\delta t}{\tau} < 2$ which can lead to instability (Mei and Shyy, 1998).

A further issue relates to the necessity – or lack thereof – of simulating fluid flows and convective heat transfer by CFD at every time step during a simulation run. An advantage of the overlap present in traditional coupling paradigms and methods is that the CFD simulation can simply be turned off when not needed or desired, allowing the convective models in an ES program to continue on their own. The ability to do this has traditionally not been an explicit feature of the Modelica language. The single assignment principle requires all variables to be computed at every time step or event and there appears to be no language level construct for the activation or inactivation of equations, families of equations, or models. Note that this concept is distinct from using `when` equations to have equations become active at certain events. Among other possibilities, recent extensions to the language might be used mimic such an effect, namely the use of the `Subtask` package and associated `mapping` annotation (The Modelica Association, 2010). Even if such a thing could be done, when using an unsteady technique such as lattice Boltzmann, the re-activation of the CFD model would require that the velocity, temperature, and heat fluxes be brought up to a state consistent with the current boundary conditions, which may have changed during the interregnum. The built-in function `reinit` may be used to achieve this however.

While turbulent flows have been simulated using Boltzmann based approaches before, usually in an LES framework, the effect of turbulence on heat transfer appears to have received little attention thus far from a practical kinetic or lattice Boltzmann viewpoint. From the macroscopic viewpoint, it has been known for some time that turbulence adds terms to the macroscopic heat flux equation, e.g. the heat flux in

direction α is given by

$$q_\alpha = -\kappa \frac{\partial \overline{T}}{\partial x_\alpha} + \rho c_p \overline{u_i' T'} \tag{5.1}$$

Here, the overbar represents a mean value and $'$ denotes a turbulent fluctuation. Equation **5.1** shows that in addition to the laminar heat flux term (the first term on the right hand side) there is also a turbulent contribution (the second term on the right hand side) (White, 2006). Derivations of the macroscopic transport and conservations equations from the Boltzmann equation have traditionally revealed the macroscopic heat flux to only have the laminar term, implying that for turbulent flow the turbulent thermal boundary layer would need to be fully resolved in a DNS type approach. Clearly this would be impractical. Future work requires investigation of this issue, which is particularly critical in light of the coupling of CFD to conduction in walls.

CHAPTER 6: CONCLUSION

6.1 Summary

Buildings are collections of many different types of systems interacting with each other; furthermore, with a few exceptions particularly in the residential sector, each building is a fairly unique design even though it may consist of standard parts. The ability to capture this uniqueness, and especially the interactions between these systems in unique configurations, is greatly facilitated by the use of physical system modeling languages. The sole purpose of a programming language is to tell a computer what to do and thus the mathematical and topological description of a system is conflated with the computational implementation of that description. Modeling languages focus on the mathematical and topological description of a system only, and in a manner more comprehendible to human modelers, leaving the computational implementation to be determined automatically by established algorithms and techniques. The mathematical description is facilitated by the use of acausal modeling in which equations, as relations among parameters and variables, are employed rather than assignment statements. The topological description is facilitated by the use of physically relevant interface constructs which are a fundamental feature of the language. Neither of these features are present in programming languages. While the hierarchical, component based paradigm of Modelica is shared with object oriented programming languages, its marriage to acausal and topological description features results in a flexible language capable of capturing unique systems configurations easily.

Because a modeling language like Modelica relies on mathematical relations to describe behavior, any behavior so describable can in principle fall under the umbrella of that language, which leads naturally to the ability to create models consisting of

many different subject domains. The presence of many subject domains in modern buildings coupled to their uniqueness suggest modeling languages to be excellent candidates for building system description. The separation between the modeling and simulation tasks has productivity benefits for those whose main concern is modeling – mathematical description of physics and topological configuration of components – relieving them of the burden of translating a model into a simulation. Furthermore the developers of Modelica tools, free from the need to have domain-specific knowledge, can focus their efforts on methods to convert models into efficient simulation executables. Modelers can remain modelers and computer scientists can remain computer scientists, a natural division of labor which permits the flexible description of systems by domain experts while leaving the problem of generating efficient simulation code to computational experts.

This study sought to demonstrate that three interacting domains present in buildings, heat transfer through conduction and convection, fluid flow, and simple controls, can be represented by the modeling language Modelica and simulated by a tool which implements Modelica models, with the goal of enabling modeling flexibility. In particular, convection and fluid flow as determined by computational fluid dynamics techniques, specifically lattice Boltzmann. The coupling between CFD and other domains has been achieved before in the context of pre-existing, monolithic building energy performance simulation tools and more traditional CFD programs based on the macroscopic equations of fluid motion, namely the continuity, Navier-Stokes, and energy equations. This study differs not only in the CFD technique employed, but also in the use of a generic modeling language and the coupling mechanism that this language implies. The use of the lattice Boltzmann method provides a relatively simple yet fully unsteady description of fluid motion and convective heat transfer, in contrast to the quasi-unsteady approaches previously studied. Furthermore lattice Boltzmann enables the simple coupling of convective heat transfer in the fluid to the conductive heat transfer in a wall since heat flux is inherently included in the model

and is a quantity determined entirely local to a node at a given timestep.

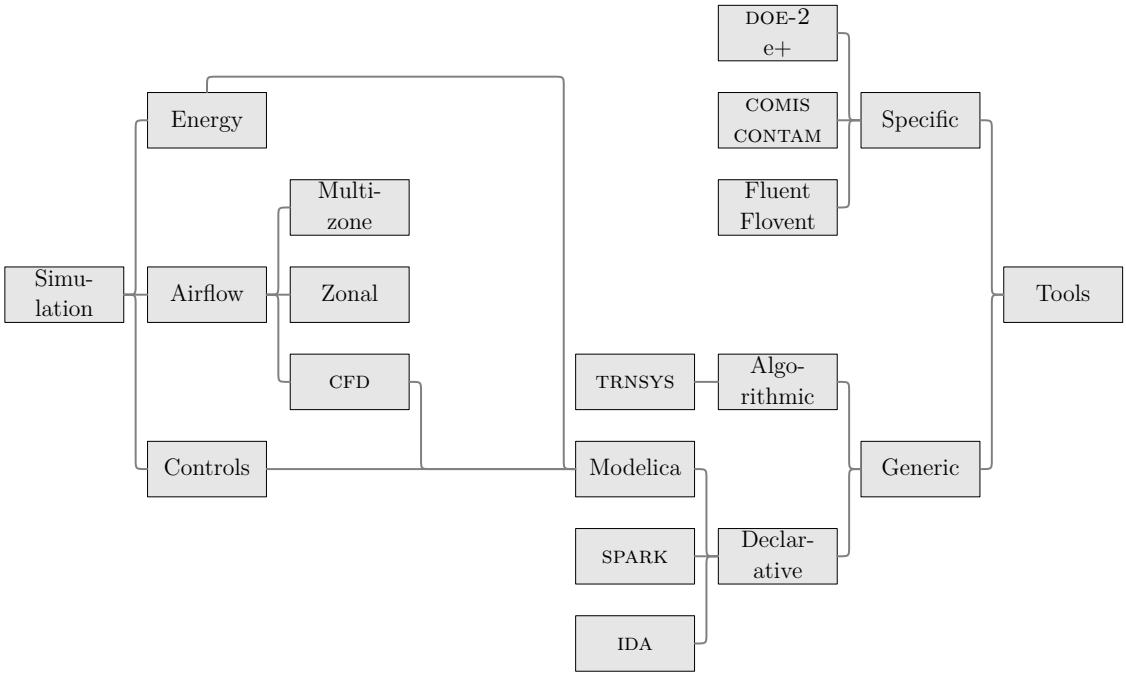


Figure 6.1.1 This Work's Contribution

These claims have been demonstrated through the initial verifications, validations, and demonstrations in Chapter 4. Models incorporating these multiple subject domains can be flexibly defined and be simulated, but whether or not the simulation is efficient is a matter of context. If the time horizon of a simulation is small and the models reflect the (small) physical situation being simulated then the models developed here are appropriate. However for larger problems with longer time horizons the appropriateness is questionable given not only the fundamental large differences in timescales between the different physical processes involved, but also given the differences in the minimum timesteps required by the models. In such cases the quasi-unsteady and overlapping aspects of previous couplings between CFD and energy simulation have immediate practical advantages over the present work, namely the ability to turn off the CFD computations during quasi-steady periods of time. However there may be ways to effect this in Modelica models as briefly mentioned in

the previous chapter. In addition Dymola, the current premiere tool for implementing Modelica models, has difficulties with CFD models of a size that hand coded programs can easily handle.

6.2 Recommendations

The recommendations given here fall under the categories of improving the efficiency of the simulations and expanding the capabilities of the models. Improving the efficiency (for example speeding up the computations) and expanding capabilities (modeling more physics and increasing the sizes of simulatable problems) can be attacked through computer science or modeling approaches. Here the modeling approaches are enumerated:

1. **Improving efficiency: capture the relevant physics using the minimum number of variables/equations**

   The lattice Boltzmann models used in this work all employ uniform lattices, while the phenomena most important to coupling to heat conduction occurs in the boundary layer. For more energetic flows, more lattice sites are required in the boundary layer although the current approaches add lattice sites throughout the CFD domain where they may not be necessary. Local grid refinement should be incorporated, in the context of either the current traditional discretization or alternative discretizations of the Boltzmann equation.

2. **Improving efficiency: create models which allow larger timesteps**

   Differences in timescales between processes may be a fundamental feature, but the difficulties that this leads to should not be compounded by the models. Decoupling the timestep from the grid size should be investigated, including the use of the method of lines to enable continuous system modeling ('continuous' in the context of the Modelica language), and alternative discretizations of both space and time for fully discrete models.

3. **Improving efficiency: take advantage of computational advances and improved language features**

   In a sense these are computer science approaches, but they should be investigated to the extent that they have been incorporated into the Modelica language or executors thereof. For example, the use of Modelica task and subtask (The Modelica Association, 2010).

4. **Improving efficiency: is CFD necessary at all times?**

   CFD may not be needed for the entire time that a model is to be simulated. Approaches to effectively turn CFD off can improve simulation efficiency during these times.

5. **Expanding the capabilities: incorporate turbulence modeling**

   Most problems of interest in buildings involve turbulent flow and any tool or model to simulate flows must at some level account for turbulence. The effect of turbulent flow on heat flux at walls should be investigated in the context of kinetic theory and Boltzmann-based CFD techniques.

6. **Expanding the capabilities: handling non-isothermal walls**

   Although non-isothermal walls have been handled in this study, an improved way to handle them is of interest. One approach that is immediately available is to group nodes together into bins and assume all nodes in a bin are at the same temperature, allowing a variable resolution capability for non-isothermal walls. However the ability to handle such situations in a more elegant and physical way through interfaces between CFD and wall conduction is a topic of interest to this author.

Together these recommendations define a large trade space and suggest many interrelated projects for the future.

APPENDIX A: THE BOLTZMANN EQUATION

Historically the lattice Boltzmann method arose as an extension and improvement of lattice gas cellular automata models of fluid flow. However it is more profitable to consider lattice Boltzmann as a child of classical statistical physics (Wolf-Gladrow, 2000) . Some fundamentals of this field are necessary background for the material in this thesis, and as this subject is unfamiliar to most engineers, a brief overview will be given in this Appendix. The discussion to follow is derived from Harris (1971), Cercignani (1988) and Liboff (1998), with supplementary citations given as noted.

The Boltzmann Equation

Consider a fluid system made up of N molecules, where N is of order 10^{23}. For simplicity, we take the case where there are no external forces such as magnetic fields acting on this system. Take these molecules to be monatomic, that is, featureless and 'rigid'; thus the notion of molecule orientation is inapplicable and a molecule cannot vibrate about its own center of mass. Any molecule $i \in \{1, 2, \cdots, N\}$ is then fully specified in space by 3 positional coordinates $\boldsymbol{x}_i(t) = (x_i(t), y_i(t), z_i(t))$, has $\eta = 3$ spatial degrees of freedom, and will be referred to in the abstract as a particle. Each particle has 3 components of momentum $\boldsymbol{p}_i(t) = (p_{x_i}(t), p_{y_i}(t), p_{z_i}(t))$ and so the particle's state is specified by the 6-component combination vector $\boldsymbol{\varsigma}_i(t) = (\boldsymbol{x}_i(t), \boldsymbol{p}_i(t))$. The bulk fluid thus has $6N$ degrees of freedom at the system level and it's microstate is fully specified by a single coordinate point $\boldsymbol{\gamma}(t) = (\boldsymbol{\varsigma}_1(t), \boldsymbol{\varsigma}_2(t), \cdots, \boldsymbol{\varsigma}_i(t), \cdots, \boldsymbol{\varsigma}_N(t))$ in the $6N$ dimensional phase space Γ.

Statistical physics operates under the claim that the intensive macroscopic properties of a system such as temperature $T(t)$, velocity field $\boldsymbol{u}(t)$, viscosity, specific heat, etc. (the macrostate), is derivable from knowledge of the microstate $\boldsymbol{\gamma}(t)$. With N

being of order 10^{23}, this is both unknowable and intractable even if knowable, but progress is possible with recourse to statistics. A fluid system can have n replicas which are each in identical macrostates but different microstates; indeed it would be remarkable if all replicas have the same microstate. Thus any macrostate corresponds to a large number of microstates, so perfect knowledge of the microstate is unnecessary. Indeed this is fortunate as we can use low-resolution (statistical) microstate information and still gain physical insight from the microstate approach.

Each of the n replicas in the ensemble is represented by one phase space coordinate point $\boldsymbol{\gamma}_k(t)$ with $k \in [1, 2, \cdots, n]$. With a large number of replicas, e.g. $n \to \infty$, these coordinate points form a dense cloud which can be characterized by a probability density function $F_N(\boldsymbol{\gamma}, t)$ so that function $nF_N(\boldsymbol{\gamma}, t)d\boldsymbol{x}d\boldsymbol{p} = nF_N(\boldsymbol{\gamma}, t)d\boldsymbol{\gamma}$ is the number of replica systems in a differential volume $d\boldsymbol{\gamma} = \prod_{i=1}^{N} d\boldsymbol{x}_i d\boldsymbol{p}_i$ about the phase point $\boldsymbol{\gamma}$. F_N thus refers to a single system whereas nF_N refers to the ensemble.

Each replica system $\boldsymbol{\gamma}_k(t)$ traces out a path, or trajectory, in phase space as time evolves. These trajectories do not cross: such a crossing means identical microstates; systems with identical microstates must evolve identically since the microstate deterministically – in classical statistical physics – evolves from initial conditions; therefore system trajectories in phase space are either coincident for all time or do not cross.

The differential phase space volume $d\boldsymbol{\gamma}$ contains phase space points (i.e. ensemble members) and is bounded by a surface $S(\boldsymbol{\gamma})$ that is made up of phase space points. As time evolves, each point traces out its own unique trajectory, therefore $d\boldsymbol{\gamma}$ and $S(\boldsymbol{\gamma})$ change shape. Because trajectories do not cross, the points inside $d\boldsymbol{\gamma}$ cannot cross $S(\boldsymbol{\gamma})$ and thus remain inside $d\boldsymbol{\gamma}$. Thus $nF_N(\boldsymbol{\gamma}, t)d\boldsymbol{\gamma}$ is constant, or

$$\frac{dF_N}{dt} = 0 = \frac{\partial F_N}{\partial t} + \sum_{i=1}^{N}\left[\frac{\partial F_N}{\partial \boldsymbol{x}_i} \cdot \dot{\boldsymbol{x}}_i + \frac{\partial F_N}{\partial \boldsymbol{p}_i} \cdot \dot{\boldsymbol{p}}_i\right] \tag{1}$$

where the overhead dot denotes the partial derivative with respect to time. The derivative dF_N/dt can be thought of as a material (a.k.a. substantive) derivative for

phase space in a Lagrangian reference frame which travels with the 'flow' of phase space points; the terms on the right hand side are in an Eulerian reference frame which is fixed in phase space, i.e. a fixed 'laboratory' frame. Equation **1** is the Liouville equation. It still can be rather intractable, however the situation can be improved using the reduced particle distribution functions

$$F_r(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \cdots, \boldsymbol{\gamma}_r, t) = \int F_N(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \cdots, \boldsymbol{\gamma}_N, t) d\boldsymbol{\gamma}_{r+1} \cdots d\boldsymbol{\gamma}_N \tag{2}$$

If $F_N d\boldsymbol{\gamma}$ is the joint probability that in a system, particle 1 is in the volume $d\boldsymbol{\gamma}_1$ about $\boldsymbol{\gamma}_1$, particle 2 is in the volume $d\boldsymbol{\gamma}_2$ about $\boldsymbol{\gamma}_2$, etc., up to the N^{th} particle, then F_r is the same quantity except it is valid only up to particle r. It turns out that the single or double particle density distribution functions F_1 and F_2 are the most important for most purposes.

Using the Liouville equation and some labor, the evolution of the reduced particle distribution functions can be expressed as the BBGKY hierarchy of equations:

$$\frac{\partial F_r}{\partial t} + \sum_{i=1}^{r} \dot{\boldsymbol{x}}_i \cdot \frac{\partial F_r}{\partial \boldsymbol{x}_i} - \sum_{i,j=1} \frac{\partial \phi_{ij}}{\partial \boldsymbol{x}_i} \cdot \frac{\partial F_r}{\partial \boldsymbol{p}_i} = (N - r) \int d\boldsymbol{\gamma}_{r+1} \sum_{i=1}^{r} \frac{\partial \phi_{ir+1}}{\partial \boldsymbol{x}_i} \cdot \frac{\partial F_{r+1}}{\partial \boldsymbol{p}_i} \tag{3}$$

where m is the mass of a particle and ϕ_{ij} is a potential for a force on the i^th particle by the j^th particle, such that the total force acting on particle i by all particles is $-\sum_{j=1\neq i}^{N} \partial \phi_{ij} / \partial \boldsymbol{x}_i$.

The equations in the hierarchy are coupled, however the first equation – for $r = 1$ – can be closed under certain mathematical limits and physical assumptions, including that the system involves only binary collisions between particles (sufficiently low density) and that particles are uncorrelated upon initiation of a collision (molecular chaos or Stosszahlansatz: any correlation that exists after a collision is short lived and dies away before another collision (Succi et al., 2002)). Together these conditions describe a perfect gas, for which the closed $R = 1$ BBGKY equation is then

$$\frac{\partial F_1}{\partial t} + \dot{\boldsymbol{x}}_1 \cdot \frac{\partial F_1}{\partial \boldsymbol{x}_1} = \Omega_B \tag{4}$$

This is the Boltzmann equation, where Ω_B is the Boltzmann collision term, an integral encoding the details of binary collisions. This integral is complex and contains nonlinear terms, and is ultimately unimportant for present purposes as a simplified version will be introduced later. This equation can be made more relatable to the macrostate by introducing the probability density function

$$\check{F}_1(\boldsymbol{\gamma}_1, t) = \check{F}_1(\boldsymbol{x}_1, \boldsymbol{p}_1, t) = NmF_1(\boldsymbol{\gamma}_1, t) \tag{5}$$

so that the quantity $\check{F}_1(\boldsymbol{\gamma}_1, t)d\boldsymbol{\gamma}_1$ is the expected mass in the differential volume $d\boldsymbol{\gamma}_1$ about $\boldsymbol{\gamma}_1$ at time t. Similarly, using $\aleph$ is the number density (density of the N particles), then

$$f_1(\boldsymbol{\gamma}_1, t) = f_1(\boldsymbol{x}_1, \boldsymbol{p}_1, t) = \aleph m F_1(\boldsymbol{\gamma}_1, t) \tag{6}$$

and f_1 is the expected mass *density* in the differential volume $d\boldsymbol{\gamma}_1$ about $\boldsymbol{\gamma}_1$ at time t.

Writing $\boldsymbol{\xi}_1 = \dot{\boldsymbol{x}}_1 = \frac{\boldsymbol{p}_1}{m}$ for the particle velocity (i.e. the 'microscopic' velocity; note that we can often simply replace $\boldsymbol{p}$ with $\boldsymbol{\xi}$) and dropping the subscript such that $\star = \star_1$, since we will be dealing exclusively with the single particle distribution, we then have:

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}} = \Omega_B \tag{7}$$

In the laboratory (Eulerian) reference frame in which differential control volumes in phase space are fixed and through which the phase space points 'flow', this equation means that the time rate of change of f, $\partial f/\partial t$, is due to particles being 'knocked' into the control volume – represented in terms of f by Ω_B – minus the advection of particles

into/out of the control volume – represented in terms of f by $\boldsymbol{\xi} \cdot \partial f / \partial \boldsymbol{x}_1 = \boldsymbol{\xi} \cdot \nabla f$ (Reichl, 1998).

Macroscopic Behavior: Some of the Moments of f

Given the definition of f, it is apparent that the macroscopic density ρ of the fluid system is

$$\rho(\boldsymbol{x}, t) = \int f(\boldsymbol{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi} \tag{8}$$

The density is sometimes referred to as the zeroth moment of f. Averaging the molecular velocities $\boldsymbol{\xi}$ yields the first moment of f, the macroscopic momentum density $\rho \boldsymbol{u}$

$$\rho(\boldsymbol{x}, t) \boldsymbol{u}(\boldsymbol{x}, t) = \int \boldsymbol{\xi} f(\boldsymbol{x}, \boldsymbol{\xi}, t) d\boldsymbol{\xi} \tag{9}$$

where $\boldsymbol{u}$ is the macroscopic fluid velocity. To consider the flux of this momentum density, we switch to tensor notation and note that the general expression for flux is given by $\int v_\alpha (v_\beta f) d\boldsymbol{v}$, where $\alpha, \beta = x, y, z$ indicate component directions. Using the macroscopic relations above and introducing the 'intrinsic' or 'peculiar' velocity $\boldsymbol{c} = \boldsymbol{\xi} - \boldsymbol{u}$, which refers to random motions of the molecules relative to the macroscopic flow, the flux of momentum density can easily be split into macroscopic and microscopic components

$$\int \xi_\alpha (\xi_\beta f) d\boldsymbol{\xi} = \rho u_\alpha u_\beta + \int c_\alpha c_\beta d\boldsymbol{\xi} \tag{10}$$

The microscopic term $\int c_\alpha c_\beta d\boldsymbol{v}$ constitutes the stress tensor $\Phi_{\alpha\beta}$ for a perfect gas, where the diagonal terms constitute normal stresses and the off-diagonal terms represent shear stresses. For other fluids $\int c_\alpha c_\beta d\boldsymbol{v}$ is one contributor to $\Phi_{\alpha\beta}$ (Schlichting (1979) has a good discussion of $\Phi_{\alpha\beta}$ from a macroscopic perspective).

Similarly considering what might be termed the second moment of f, and anticipating this moment to be an expression of kinetic energy, we can write

$$\frac{1}{2}\int \xi^2 f d\boldsymbol{\xi} = \frac{1}{2}\rho u^2 + \frac{1}{2}\int c^2 f d\boldsymbol{\xi} \tag{11}$$

Here $\frac{1}{2}\rho u^2$ represents the kinetic energy density of the macroscopic flow while the peculiar microscopic kinetic energy $\frac{1}{2}\int c^2 f d\boldsymbol{\xi}$ can intuitively be seen as the internal energy density per unit volume $\rho\epsilon$. Incorporating the equipartition theorem (Sturge, 2003), which states that the internal energy ϵ for a system of $\approx 10^{23}$ particles is $\frac{1}{2}\eta\frac{N_A k}{m_{mol}}T$, where N_A is Avogadro's number, k is Boltzmann's constant, and m_{mol} is the molar mass, then we can write equation **11** as

$$\frac{1}{2}\int \xi^2 f d\boldsymbol{\xi} = \frac{1}{2}\rho u^2 + \frac{1}{2}\rho\eta RT \tag{12}$$

where $R = \frac{N_A k}{m_{mol}}$ is the specific gas constant.

The term $\frac{1}{2}\int c^2 f d\boldsymbol{\xi}$ also bears a resemblance to the trace of the stress tensor. If we consider the case of a system in static equilibrium – the fluid is macroscopically at rest – then the normal stresses are all equal and thus the trace of the stress tensor $\frac{1}{3}\Phi_{\alpha\beta}$ can be identified as the hydrostatic pressure p. Once again using the equipartition theorem,

$$p = \frac{1}{3}\Phi_{\alpha\alpha} = \frac{2}{3}\left[\rho\frac{1}{2}\int c^2 f d\boldsymbol{v}\right] \tag{13}$$

$$= \frac{2}{3}\left[\rho\frac{1}{2}\eta N_A k T\right] \tag{14}$$

$$= \rho RT \tag{15}$$

where $\eta = 3$ for a monatomic gas. Thus through theoretical reasoning we have derived the ideal (perfect) gas equation of state.

Again using the general expression for flux, applied now to energy (the second moment), and using the relations given thus far along with some supplementary relations regarding the peculiar velocity, the flux of energy can be written

$$\int \xi_\alpha \left(\frac{1}{2}\xi^2 f\right) d\boldsymbol{\xi} = \xi_\alpha \left(\frac{1}{2}\rho u^2 + \frac{1}{2\rho}\int c^2 f d\boldsymbol{\xi}\right) + \xi_\beta \Phi_{\alpha\beta} + \frac{1}{2}\int c_\alpha c^2 f d\boldsymbol{\xi} \tag{16}$$

The terms here represent the convection of macro- and microscopic energy by the macroscopic velocity, a mechanical work term, and the transport of microscopic energy by microscopic motion; this last term can be seen as the microscopic flux of internal energy and thus is manifested on the macroscopic scale as thermal conduction (heat flux) and is therefore labeled as $q_\alpha = \frac{1}{2}\int c_\alpha c^2 f d\boldsymbol{\xi}$.

Macroscopic Behavior: Conservation equations

The basic forms of the macroscopic hydrodynamic conservation equations can be derived from the Boltzmann equation or a modified form thereof using the preceding macroscopic relations, and a few other basic relations which for purposes of brevity are neglected here. The raw Boltzmann equation can be worked into the general continuity equation for a compressible fluid

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_\alpha}(\rho u_\alpha) = 0 \tag{17}$$

which should not be surprising since the Boltzmann equation is derived starting from an expression for the conservation of the 'fluid' of phase space points. Multiplying the Boltzmann equation by $\boldsymbol{u}$ and applying some labor yields the elementary form of the conservation of momentum where body forces are neglected

$$\frac{\partial}{\partial t}(\rho u_\beta) + \frac{\partial}{\partial x_\alpha}(\rho u_\alpha u_\beta + \Phi_{\alpha\beta}) = 0 \tag{18}$$

Multiplying the Boltzmann equation by $\frac{1}{2}v^2$ and working again without body forces gives an energy equation

$$\frac{\partial}{\partial t}\left[\rho\left(\frac{1}{2}u^2+\frac{1}{2\rho}\int c^2 f d\boldsymbol{v}\right)\right]+\frac{\partial}{\partial x_\alpha}\left[\rho u_\alpha\left(\frac{1}{2}u^2+\frac{1}{2\rho}\int c^2 f d\boldsymbol{v}\right)+\Phi_{\alpha\beta}u_\beta+q_\alpha\right]=0 \tag{19}$$

The preceding two equations can be closed using constitutive equations. For an isotropic Newtonian, conducting fluid these are :

$$\Phi_{\alpha\beta}=p\delta_{\alpha\beta}-\lambda\left(\frac{\partial x_\gamma}{\partial u_\gamma}\right)\delta_{\alpha\beta}-\mu\left(\frac{\partial u_\alpha}{\partial x_\beta}+\frac{\partial u_\beta}{\partial x_\alpha}\right) \tag{20}$$

$$q_\alpha=\kappa\frac{\partial T}{\partial x_\alpha} \tag{21}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta, λ is the second viscosity coefficient, μ is the dynamic viscosity, and κ is the thermal conductivity. Notice that for a any fluid in global equilibrium (macroscopically at rest or free of velocity gradients), the trace of the stress tensor indeed is the thermodynamic pressure, as previously noted. For an inviscid fluid this is also the case under nonequilibrium conditions, while for viscous fluids the trace of the stress tensor contains the pressure as one contributor. Under Stokes' hypothesis, one takes $\lambda=-\frac{2}{3}\mu$ and the bulk viscosity $\mu\prime=\lambda-\frac{2}{3}\mu$ vanishes; this is the case gasses, and is likewise applicable for liquids in chemical equilibrium (incompressibility renders the question moot) (Schlichting, 1979). For an incompressible fluid, $\frac{\partial x_\gamma}{\partial u_\gamma}=0$ and the term involving λ disappears. When these constitutive equations applied to the general momentum and energy conservation equations, the Navier-Stokes and a version of the macroscopic energy equation are produced.

The progression from a microscopic description to the macroscopic continuity, Navier-Stokes, and energy equations is given here only to show the correctness of the microscopic approach to those more familiar with the macroscopic approach. The microscopic approach in fact does not require constitutive equations or experimentally

determined transport coefficients such as μ and κ, as all necessary information for any macroscopic property or phenomena, at least for a perfect gas, is available through f and the Boltzmann equation.

Up to this point much has been gained without any knowledge of the form of f or any actual solution to the Boltzmann equation, but further progress requires the examination of both.

The Equilibrium Distribution and a Simplified Collision Term

Solutions to the Boltzmann equation yield values or expressions for f, however the solution of the full Boltzmann equation is difficult. Nevertheless, the form of f for an equilibrium state, f^{EQ}, can be determined by considering the manner in which a system of particles approaches equilibrium. The H theorem states that the quantity

$$H = \int f ln f d\boldsymbol{\xi} \tag{22}$$

decreases as a system evolves toward an equilibrium state, and that this decrease in H is bounded. Further $\partial H/\partial t = 0$ at equilibrium. An argument, omitted here as it deals with the full collision integral Ω_B and other details likewise omitted, yields for the for the equilibrium distribution

$$f^{EQ} = \frac{\rho}{(2\pi RT)^{(\eta/2)}} e^{\frac{-(\boldsymbol{\xi}-\boldsymbol{u})^2}{2RT}} \tag{23}$$

which is termed the Maxwell-Boltzmann, or Maxwellian, distribution.

The complexity of the collision term Ω_B precludes the use of the Boltzmann equation in many practical problems, but a vastly simplified form which retains essential physics appropriate for many cases was developed by Bhatnagar et al. (1954), which leads to a simplified Boltzmann equation

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \frac{\partial f}{\partial \boldsymbol{x}_1} = \frac{1}{\theta}(f^{EQ} - f) \tag{24}$$

where θ is the time to relax to equilibrium. Equations **23** and **24** form the basis of the lattice-Boltzmann method.

APPENDIX B: SAMPLE MODELICA MODELS

Some representative examples of the models used in this work are given here. The text here has been edited for brevity and to fit on the page: repetitive sections and non-essential commentary, etc., are removed.

Sample node models

Partial Node

```
partial model PrtlNode
 public
  // parameters and constants
  outer DomainParameters DmPa;
  //
  // hydrodynamic variables
  // these are initialized in an initial equation section of a CFD model (e.g. see "PlanarCouette")
  outer discrete SIu.Pressure   p[DmPa.Num_ry, DmPa.Num_cx]  "Pressure; Pa";
  outer discrete SIu.Velocity    u[DmPa.Num_ry, DmPa.Num_cx]  "Velocity, x-direction; m/s";
  outer discrete SIu.Velocity   v[DmPa.Num_ry, DmPa.Num_cx]  "Velocity, y-direction; m/s";
  outer discrete SIu.Temperature  T[DmPa.Num_ry, DmPa.Num_cx]  "Temperature; K";
  //
  // x-location is DmPa.x[cx]; cx = column; y-location is DmPa.y[cx]; ry = row
  // cx and ry are assigned when nodes are assembeled into domain (see the domain models, e.g. "GenericDomain")
  parameter Integer cx ;//annotation(HideResult=true);
  parameter Integer ry ;//annotation(HideResult=true);
  //
  Boolean clock   annotation(HideResult=true);
  //
 protected
  // single particle distributions
  outer discrete lbMassDistro    f[DmPa.Num_ry, DmPa.Num_cx,8] annotation(HideResult=true);
  outer discrete lbMassDistro  feq[DmPa.Num_ry, DmPa.Num_cx,8] annotation(HideResult=true);
  outer discrete lbMassDistro   f0[DmPa.Num_ry, DmPa.Num_cx]  annotation(HideResult=true);
  outer discrete lbMassDistro feq0[DmPa.Num_ry, DmPa.Num_cx]  annotation(HideResult=true);
  //
  // internal energy distributions
  outer discrete lbEnergyDistro   gm[DmPa.Num_ry, DmPa.Num_cx,8] annotation(HideResult=true);
  outer discrete lbEnergyDistro  geq[DmPa.Num_ry, DmPa.Num_cx,8] annotation(HideResult=true);
  outer discrete lbEnergyDistro  g0m[DmPa.Num_ry, DmPa.Num_cx]  annotation(HideResult=true);
  outer discrete lbEnergyDistro geq0[DmPa.Num_ry, DmPa.Num_cx]  annotation(HideResult=true);
  //
  // molecular velocities
  parameter SIu.Velocity xix[8] = {DmPa.c, 0,    -DmPa.c,   0,   DmPa.c, -DmPa.c, -DmPa.c,  DmPa.c};
  parameter SIu.Velocity xiy[8] = {  0,  DmPa.c,    0,   -DmPa.c, DmPa.c,  DmPa.c, -DmPa.c, -DmPa.c};
  //
  // normailized molecular velocities: for use in array indexing
  constant Integer xinx[8] = {1, 0, -1,   0,  1,  -1, -1,  1}  annotation(HideResult=true);
  constant Integer xiny[8] = {0, 1, 0, -1,  1,   1, -1, -1}  annotation(HideResult=true);
initial equation
 // the single particle equilibrium distributions
 feq0[ry,cx]  = DmPa.rho0 - (20/12)*p[ry,cx]/DmPa.c^2 - DmPa.rho0*( (2/3)*(u[ry,cx]^2 + v[ry,cx]^2)/DmPa.c^2 );
 feq[ry,cx,1] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[1]*u[ry,cx] + xiy[1]*v[ry,cx])/DmPa.c^2 + ...;
 feq[ry,cx,2] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[2]*u[ry,cx] + xiy[2]*v[ry,cx])/DmPa.c^2 + ...;
 feq[ry,cx,3] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[3]*u[ry,cx] + xiy[3]*v[ry,cx])/DmPa.c^2 + ...;
 feq[ry,cx,4] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[4]*u[ry,cx] + xiy[4]*v[ry,cx])/DmPa.c^2 + ...;
 feq[ry,cx,5] = (1/12)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*( (1/12)*(xix[5]*u[ry,cx] + xiy[5]*v[ry,cx])/DmPa.c^2 + ...;
 ...
 //
 // the internal energy equilibrium distributions
```

```
 geq0[ry,cx]  = DmPa.rho0*DmPa.R*T[ry,cx]*(  (4/9) - (2/3)*(u[ry,cx]^2 + v[ry,cx]^2)/DmPa.c^2 );
 geq[ry,cx,1] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[1]*u[ry,cx] + xiy[1]*v[ry,cx])/DmPa.c^2 + ...;
 geq[ry,cx,2] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[2]*u[ry,cx] + xiy[2]*v[ry,cx])/DmPa.c^2 + ...;
 geq[ry,cx,3] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[3]*u[ry,cx] + xiy[3]*v[ry,cx])/DmPa.c^2 + ...;
 geq[ry,cx,4] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[4]*u[ry,cx] + xiy[4]*v[ry,cx])/DmPa.c^2 + ...;
 geq[ry,cx,5] = DmPa.rho0*DmPa.R*T[ry,cx]*( (1/36) + (1/12)*(xix[5]*u[ry,cx] + xiy[5]*v[ry,cx])/DmPa.c^2 + ...;
 ...
 //
 // initialize f and g with feq and geq
 f0[ry,cx] = feq0[ry,cx];
 g0m[ry,cx] = geq0[ry,cx];
 for k in 1:8 loop
  f[ry,cx,k] = feq[ry,cx,k];
  gm[ry,cx,k] = geq[ry,cx,k];
 end for;
 //
equation
 clock = sample(0, DmPa.delt);
 when {clock} then
  // the single particle equilibrium distributions
  feq0[ry,cx]  = DmPa.rho0 - (20/12)*p[ry,cx]/DmPa.c^2 - DmPa.rho0*( (2/3)*(u[ry,cx]^2 + v[ry,cx]^2)/DmPa.c^2
);
  feq[ry,cx,1] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[1]*u[ry,cx] + xiy[1]*v[ry,cx])/DmPa.c^2 +
...;
  feq[ry,cx,2] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[2]*u[ry,cx] + xiy[2]*v[ry,cx])/DmPa.c^2 +
...;
  feq[ry,cx,3] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[3]*u[ry,cx] + xiy[3]*v[ry,cx])/DmPa.c^2 +
...;
  feq[ry,cx,4] =  (1/3)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*(  (1/3)*(xix[4]*u[ry,cx] + xiy[4]*v[ry,cx])/DmPa.c^2 +
...;
  feq[ry,cx,5] = (1/12)*p[ry,cx]/DmPa.c^2 + DmPa.rho0*( (1/12)*(xix[5]*u[ry,cx] + xiy[5]*v[ry,cx])/DmPa.c^2 +
...;
  ...
  //
  // the internal energy equilibrium distributions
  geq0[ry,cx]  = DmPa.rho0*DmPa.R*T[ry,cx]*(  (4/9) - (2/3)*(u[ry,cx]^2 + v[ry,cx]^2)/DmPa.c^2 );
  geq[ry,cx,1] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[1]*u[ry,cx] + xiy[1]*v[ry,cx])/DmPa.c^2 + ...;
  geq[ry,cx,2] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[2]*u[ry,cx] + xiy[2]*v[ry,cx])/DmPa.c^2 + ...;
  geq[ry,cx,3] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[3]*u[ry,cx] + xiy[3]*v[ry,cx])/DmPa.c^2 + ...;
  geq[ry,cx,4] = DmPa.rho0*DmPa.R*T[ry,cx]*(  (1/9) +  (1/3)*(xix[4]*u[ry,cx] + xiy[4]*v[ry,cx])/DmPa.c^2 + ...;
  geq[ry,cx,5] = DmPa.rho0*DmPa.R*T[ry,cx]*( (1/36) + (1/12)*(xix[5]*u[ry,cx] + xiy[5]*v[ry,cx])/DmPa.c^2 + ...;
  ...

uot; end when;
end PrtlNode;
```

Partial Fluid Node

```
partial model PrtlNodeFluid
 extends PrtlNode;
 // adds the following:
 // 1. streaming + collision for the rest distributions f0 and g0
 // 2. computes the moments of f (velocity and pressure) and gm (internal energy density/temperature)
equation
 when {clock} then
  //
  // streaming + collision for the rest distributions (are the same for all nodes);
  //  non-rest distros handled in specialized node models
  f0[ry,cx] = pre(f0[ry,cx]) - (1/DmPa.tau_u)*( pre(f0[ry,cx]) - pre(feq0[ry,cx]) )
        - DmPa.delt*DmPa.rho0*(DmPa.beta)*
                        ( pre(T[ry,cx]) - DmPa.T0 )*
                        pre(feq0[ry,cx])*
                        ( DmPa.grav[1]*(-1)*pre(u[ry,cx]) + DmPa.grav[2]*(-1)*pre(v[ry,cx]) )/
                        pre(p[ry,cx]);
  //
  g0m[ry,cx] = pre(g0m[ry,cx]) - (1/(DmPa.tau_e+0.5))*( pre(g0m[ry,cx]) - pre(geq0[ry,cx]) );

uot;  //
  // the moments of f and g
  u[ry,cx] = (1/DmPa.rho0)*(  xix[1]*f[ry,cx,1] + xix[2]*f[ry,cx,2] + xix[3]*f[ry,cx,3] + xix[4]*f[ry,cx,4]
                 + xix[5]*f[ry,cx,5] + xix[6]*f[ry,cx,6] + xix[7]*f[ry,cx,7] + xix[8]*f[ry,cx,8]
               );
  v[ry,cx] = (1/DmPa.rho0)*(  xiy[1]*f[ry,cx,1] + xiy[2]*f[ry,cx,2] + xiy[3]*f[ry,cx,3] + xiy[4]*f[ry,cx,4]
              + xiy[5]*f[ry,cx,5] + xiy[6]*f[ry,cx,6] + xiy[7]*f[ry,cx,7] + xiy[8]*f[ry,cx,8]
```

```
              );
  p[ry,cx] = (12/20)*(DmPa.c^2)*(   f[ry,cx,1] + f[ry,cx,2] + f[ry,cx,3] + f[ry,cx,4]
                  + f[ry,cx,5] + f[ry,cx,6] + f[ry,cx,7] + f[ry,cx,8]
                  - DmPa.rho0*(2/3)*( (u[ry,cx]^2 + v[ry,cx]^2)/DmPa.c^2 )
                );
  T[ry,cx] = 1/(DmPa.rho0*DmPa.R)*( g0m[ry,cx] + gm[ry,cx,1] + gm[ry,cx,2] + gm[ry,cx,3] + gm[ry,cx,4]
                               + gm[ry,cx,5] + gm[ry,cx,6] + gm[ry,cx,7] + gm[ry,cx,8]
                  );
 end when;
end PrtlNodeFluid;
```

Fluid Node

```
model NodeFluidCore
 extends PrtlNodeFluid;
equation
 when {clock} then
  // collision + streaming in one equation
  for k in 1:8 loop
   gm[ry,cx,k] = pre(gm[ry+xiny[k], cx-xinx[k], k]) +
         (1/(DmPa.tau_e+0.5))*( pre(geq[ry+xiny[k], cx-xinx[k], k]) - pre(  gm[ry+xiny[k], cx-xinx[k], k]) );
   f[ry,cx,k] = pre(f[ry+xiny[k],cx-xinx[k],k]) + (1/DmPa.tau_u)*(pre(feq[ry+xiny[k],cx-xinx[k],k])-
   pre(f[ry+xiny[k],cx-xinx[k], k])) -
         DmPa.delt*DmPa.rho0*(DmPa.beta)*( pre(T[ry,cx]) - DmPa.T0 )*pre(feq[ry,cx,k])*( DmPa.grav[1]*(xix[k]-
pre(u[ry,cx])) +
         DmPa.grav[2]*(xiy[k]-pre(v[ry,cx])) )/pre(p[ry,cx]);
  end for;
 end when;
end NodeFluidCore;
```

Periodic Node

```
model NodeFluidEdgePeriodic  "Not inclucing corner nodes"
 extends PrtlNodeFluid;
equation
 when {clock} then
  // collision + streaming: needs to be a special kind for a given location
  //
  // if this node is on the left
  if cx==1 then
   // normal streaming for these
   f[ry,cx,2] = pre(f[ry+xiny[2], cx-xinx[2], 2]) + ...;
   f[ry,cx,6] = pre(f[ry+xiny[6], cx-xinx[6], 6]) + ...;
   f[ry,cx,3] = pre(f[ry+xiny[3], cx-xinx[3], 3]) + ...;
   f[ry,cx,7] = pre(f[ry+xiny[7], cx-xinx[7], 7]) + ...;
   f[ry,cx,4] = pre(f[ry+xiny[4], cx-xinx[4], 4]) + ...;
   //
   gm[ry,cx,2] = pre(gm[ry+xiny[2], cx-xinx[2], 2]) + ...;
   gm[ry,cx,6] = pre(gm[ry+xiny[6], cx-xinx[6], 6]) + ...;
   gm[ry,cx,3] = pre(gm[ry+xiny[3], cx-xinx[3], 3]) + ...;
   gm[ry,cx,7] = pre(gm[ry+xiny[7], cx-xinx[7], 7]) + ...;
   gm[ry,cx,4] = pre(gm[ry+xiny[4], cx-xinx[4], 4]) + ...;
   //
   // these need to reach around to the other side
   f[ry,cx,5] = f[ry, DmPa.Num_cx, 5];
   f[ry,cx,1] = f[ry, DmPa.Num_cx, 1];
   f[ry,cx,8] = f[ry, DmPa.Num_cx, 8];
   //
   gm[ry,cx,5] = gm[ry, DmPa.Num_cx, 5];
   gm[ry,cx,1] = gm[ry, DmPa.Num_cx, 1];
   gm[ry,cx,8] = gm[ry, DmPa.Num_cx, 8];
  //
  //
  // if this node is on the right
  elseif cx==DmPa.Num_cx then
   // normal streaming for these
   ...
   //
   // these need to reach around to the other side
   ...
  //
  // other locations
```

```
  elseif ...
  ...
  //
  // this node is probably not placed correctly
  else
   //
   f[ry,cx,1] = 0;
   f[ry,cx,2] = 0;
   f[ry,cx,3] = 0;
   f[ry,cx,4] = 0;
   f[ry,cx,5] = 0;
   f[ry,cx,6] = 0;
   f[ry,cx,7] = 0;
   f[ry,cx,8] = 0;
   //
   gm[ry,cx,1] = 0;
   gm[ry,cx,2] = 0;
   gm[ry,cx,3] = 0;
   gm[ry,cx,4] = 0;
   gm[ry,cx,5] = 0;
   gm[ry,cx,6] = 0;
   gm[ry,cx,7] = 0;
   gm[ry,cx,8] = 0;
   //
   terminate("One or more periodic node(s) is probably not placed correctly");
  end if;
 end when;
end NodeFluidEdgePeriodic;
```

Partial Wall Node

```
model PrtlNodeEdgeWall
 extends PrtlNode;
 parameter SIu.Acceleration wallGrav[2] = {0, -9.81}  "...; m/s^2";
 parameter Integer UnitNormVec[2]     "Unit normal vector, pointing inward toward the fluid using (x,y) coord.
system";
 parameter String  PositionAndCondition = Functions.SetPosAndCond(UnitNormVec, ry, cx, DmPa.Num_ry, DmPa.Num_cx)
"..."; annotation(HideResult=true);
 annotation (Documentation(info="..."));
equation
 when {clock} then
  // streaming + collision for the rest distributions (are the same for all nodes);
  //  non-rest distros handled in specialized node models
  f0[ry,cx] = pre(f0[ry,cx]) - (1/DmPa.tau_u)*( pre(f0[ry,cx]) - pre(feq0[ry,cx]) ) -
         DmPa.delt*DmPa.rho0*(DmPa.beta)*(pre(T[ry,cx]) - DmPa.T0)*pre(feq0[ry,cx])*(wallGrav[1]*(-1)*pre(u[ry,cx])
+
         wallGrav[2]*(-1)*pre(v[ry,cx]) )/pre(p[ry,cx]);
  //
  g0m[ry,cx] = pre(g0m[ry,cx]) - (1/(DmPa.tau_e+0.5))*( pre(g0m[ry,cx]) - pre(geq0[ry,cx]) );
 end when;
end PrtlNodeEdgeWall;
```

Sample Dirichlet Node

```
 model NodeEdgeWallDirichletUnmoving  "For temperature specified on a fixed boundary"
  extends PrtlNodeEdgeWall;
  public
   BldgPhysics.Connectors.connectorTemperature connTemp;
   discrete SIu.HeatFlux q      "Heat flux rate normal to the wall; W/m^2 using cartesian (x,y) sign conventions";
   discrete Heat2D    Q2D     "2D (conserved) heat per unit depth; W/m in cartesian (x,y) sign conventions";
   annotation (Documentation(info="..."));
 equation
  when clock then
   // this is an unmoving wall
   u[ry,cx] = 0;
   v[ry,cx] = 0;
   T[ry,cx] = connTemp.T;
   //
   // run though the different nodes:
   //  nodes on walls      U, B, L, R
   //  MidWall domain corners    ULmw, URmw, BLmw, BRmw
   // TrueCorner domain corners ULrc, URrc, BLrc, BRrc
```

```
   //
   // nodes on the walls in the middle of a domain edge: U, B, L, R
   if PositionAndCondition == "U" then
    assert(UnitNormVec[1]==0 and UnitNormVec[2]==-1, "NodeEdgeWallDirichletUnmoving: UnitNormVec improperly defined
at ...");
    // the unknown populations are 4, 7, and 8
    for k in 1:8 loop
     if k==4 then
       f[ry,cx,4] = [...non-equilibruium extrapolation...]
      gm[ry,cx,4] = [...non-equilibruium bounceback...]
     elseif k==7 then
       f[ry,cx,7] = [...non-equilibruium extrapolation...]
      gm[ry,cx,7] = [...non-equilibruium bounceback...]
     elseif k==8 then
       f[ry,cx,8] = [...non-equilibruium extrapolation...]
      gm[ry,cx,8] = [...non-equilibruium bounceback...]
     else
      // these stream and collide as usual and are thus 'known'
       f[ry,cx,k] = [...stream and collide...]
      gm[ry,cx,k] = [...stream and collide...]
     end if;
    end for;
    p[ry,cx]   = (12/20)*(DmPa.c^2)*( f[ry,cx,1] + f[ry,cx,2] + f[ry,cx,3] + f[ry,cx,4] + f[ry,cx,5] + f[ry,cx,6]
+ f[ry,cx,7] + f[ry,cx,8] );
    // heat flux is in the y-direction
    q=(DmPa.tau_e/(DmPa.tau_e+0.5))*((-v[ry,cx])*g0m[ry,cx]+(xiy[1]-v[ry,cx])*gm[ry,cx,1]
    +(xiy[2]-v[ry,cx])*gm[ry,cx,2] + (xiy[3]-v[ry,cx])*gm[ry,cx,3] + (xiy[4]-v[ry,cx])*gm[ry,cx,4]
    + (xiy[5]-v[ry,cx])*gm[ry,cx,5] + (xiy[6]-v[ry,cx])*gm[ry,cx,6] + (xiy[7]-v[ry,cx])*gm[ry,cx,7]
    + (xiy[8]-v[ry,cx])*gm[ry,cx,8] - DmPa.rho0*DmPa.R*T[ry,cx]*v[ry,cx] );
    Q2D = q*(DmPa.DomainBreadth/(DmPa.Num_cx-1));
   //
   //
   elseif PositionAndCondition == "B" then
    ...
   //
   [...other positions...]
   ...
   // something's wrong
   else
    for k in 1:8 loop
     gm[ry,cx,k] = 0;
     f[ry,cx,k] = 0;
    end for;
    p[ry,cx] = DmPa.rho0*( 2-(T[ry,cx]/DmPa.T0) )*DmPa.R*DmPa.T0;
    q = 0;
    Q2D = 0;
    terminate("oops: something's wrong with NodeEdgeWallDirichletUnmoving");
   end if;
  //
  end when;
 end NodeEdgeWallDirichletUnmoving;
```

Robin Node

```
 model NodeEdgeWallRobinUnmoving   "For 'conjugation' to other models; boundary conditions are 'further out'"
  extends PrtlNodeEdgeWall;
  public
   BldgPhysics.Connectors.connectorHeat2DTemperature  connThermo;
   discrete SIu.HeatFlux q  "Heat flux rate normal to the wall; W/m^2";
   annotation (Documentation(info="..."));
 equation
  when clock then
   // this is an unmoving wall
   // P2T: can simplify the equations below to eliminate u and v
   u[ry,cx] = 0;
   v[ry,cx] = 0;
   T[ry,cx] = connThermo.T;
   // connThermo.Q2D connections are made in the code below
   //
   // run though the different nodes:
   //  nodes on walls      U, B, L, R
   //  MidWall domain corners    ULmw, URmw, BLmw, BRmw
   // TrueCorner domain corners ULrc, URrc, BLrc, BRrc
   //
```

```
   // nodes on the walls in the middle of a domain edge: U, B, L, R
   if PositionAndCondition == "U" then
    assert(UnitNormVec[1]==0 and UnitNormVec[2]==-1, "NodeEdgeWallRobinUnmoving: UnitNormVec improperly defined
at ...");
    // the unknown populations are 4, 7, and 8
    for k in 1:8 loop
     if k==4 then
        f[ry,cx,4] = [...non-equilibruium extrapolation...]
      gm[ry,cx,4] = [...non-equilibruium bounceback...]
     elseif k==7 then
       f[ry,cx,7] = [...non-equilibruium extrapolation...]
      gm[ry,cx,7] = [...non-equilibruium bounceback...]
     elseif k==8 then
       f[ry,cx,8] = [...non-equilibruium extrapolation...]
      gm[ry,cx,8] = [...non-equilibruium bounceback...]
     else
      // these stream and collide as usual and are thus 'known'
       f[ry,cx,k] = [...stream and collide...]
      gm[ry,cx,k] = [...stream and collide...]
     end if;
    end for;
    //
    p[ry,cx] = (12/20)*(DmPa.c^2)*( f[ry,cx,1] + f[ry,cx,2] + f[ry,cx,3] + f[ry,cx,4] + f[ry,cx,5] + f[ry,cx,6]
+ f[ry,cx,7] + f[ry,cx,8] );
    // heat flux is in the y-direction
    q = (DmPa.tau_e/(DmPa.tau_e+0.5))*((-v[ry,cx])*g0m[ry,cx] + ...;
    connThermo.Q2D = -q*(DmPa.DomainBreadth/(DmPa.Num_cx-1));
   //
   elseif PositionAndCondition == "B" then
   ...
   // something's wrong
   else
    for k in 1:8 loop
     gm[ry,cx,k] = 0;
     f[ry,cx,k] = 0;
    end for;
    p[ry,cx] = 0;
    q = 0;
    connThermo.Q2D = q*0;
    terminate("oops: something's wrong with NodeEdgeWallRobinUnmoving");
   end if;
  end when;
 end NodeEdgeWallRobinUnmoving;
```

Sample CFD domain models

Generic Domain

```
model DomainGeneric
 public
  // parameters and constants
  inner DomainParameters DmPa;
  //
  // hydrodynamic variables
  // these are initialized in an initial equation section of a CFD model (e.g. see "PlanarCouette")
  inner discrete SIu.Pressure   p[DmPa.Num_ry, DmPa.Num_cx]  "Pressure; Pa";
  inner discrete SIu.Velocity    u[DmPa.Num_ry, DmPa.Num_cx]  "Velocity, x-direction; m/s";
  inner discrete SIu.Velocity   v[DmPa.Num_ry, DmPa.Num_cx]  "Velocity, y-direction; m/s";
  inner discrete SIu.Temperature  T[DmPa.Num_ry, DmPa.Num_cx]  "Temperature; K";
  //
 protected
  // single particle distributions
  inner discrete lbMassDistro    f[DmPa.Num_ry, DmPa.Num_cx,8] ;//annotation(HideResult=true);
  inner discrete lbMassDistro  feq[DmPa.Num_ry, DmPa.Num_cx,8] ;//annotation(HideResult=true);
  inner discrete lbMassDistro   f0[DmPa.Num_ry, DmPa.Num_cx]  ;//annotation(HideResult=true);
  inner discrete lbMassDistro feq0[DmPa.Num_ry, DmPa.Num_cx]  ;//annotation(HideResult=true);
  //
  // internal energy distributions
  inner discrete lbEnergyDistro    gm[DmPa.Num_ry, DmPa.Num_cx,8] ;//annotation(HideResult=true);
  inner discrete lbEnergyDistro  geq[DmPa.Num_ry, DmPa.Num_cx,8] ;//annotation(HideResult=true);
  inner discrete lbEnergyDistro   g0m[DmPa.Num_ry, DmPa.Num_cx]  ;//annotation(HideResult=true);
  inner discrete lbEnergyDistro geq0[DmPa.Num_ry, DmPa.Num_cx]  ;//annotation(HideResult=true);
```

```
  //
  // arrays to locate the nodes
  parameter Integer CXarray[DmPa.Num_ry-2, DmPa.Num_cx-2] = Functions.FormCXarray(DmPa.Num_ry, DmPa.Num_cx);
  parameter Integer RYarray[DmPa.Num_ry-2, DmPa.Num_cx-2] = Functions.FormRYarray(DmPa.Num_ry, DmPa.Num_cx);
  //
  // core nodes; initialize the ry/cx members of the nodes so they 'know where they are'
  NodeFluidCore CoreNodes[DmPa.Num_ry-2, DmPa.Num_cx-2](ry=RYarray, cx=CXarray);
  //
end DomainGeneric;
```

Planar Couette

```
model DomainDirichletPlanarCouette
 extends DomainGeneric;
 public
  // boundary nodes: upper, including corners
  NodeEdgeWallDirichletMovingUpper ULCornerNode(ry=1, cx=1, UnitNormVec={0,-1});
  NodeEdgeWallDirichletMovingUpper UpperNodes[DmPa.Num_cx-2](each ry=1, cx=CXarray[1,:], each UnitNormVec={0,-
1});
  NodeEdgeWallDirichletMovingUpper URCornerNode(ry=1, cx=DmPa.Num_cx, UnitNormVec={0,-1});
  //
  // boundary nodes: bottom, including corners
  NodeEdgeWallDirichletUnmoving  BLCornerNode(ry=DmPa.Num_ry, cx=1, UnitNormVec={0,1});
  NodeEdgeWallDirichletUnmoving  BottomNodes[DmPa.Num_cx-2](each ry=DmPa.Num_ry, cx=CXarray[1,:],
   each UnitNormVec={0,1});
  NodeEdgeWallDirichletUnmoving  BRCornerNode(ry=DmPa.Num_ry, cx=DmPa.Num_cx, UnitNormVec={0,1});
  //
  // boundary nodes: the (periodic) sides
  NodeFluidEdgePeriodic   LeftNodes[DmPa.Num_ry-2](ry=2:DmPa.Num_ry-1, each cx=1);
  NodeFluidEdgePeriodic  RightNodes[DmPa.Num_ry-2](ry=2:DmPa.Num_ry-1, each cx=DmPa.Num_cx);
end DomainDirichletPlanarCouette;
```

Sample Interface

This model interfaces between a CFD domain and other models, such as conduction.

```
model ConjugateInterfaceIsothermal
 parameter Integer Num_nodes;
 BldgPhysics.Connectors.connectorHeat2DTemperature connWallSideThermo;
 BldgPhysics.Connectors.connectorHeat2DTemperature connFluidSideThermo[Num_nodes];
equation
 for j in 1:Num_nodes loop
  connect(connWallSideThermo, connFluidSideThermo[j]);
 end for;
end ConjugateInterfaceIsothermal;
```

Sample heat transfer models

```
model Conduction1D_HomoIsoWall  "One dimensional conduction for homogeneous isotropic wall"
 BldgPhysics.Connectors.connectorHeat2DTemperature connThermoA;
 BldgPhysics.Connectors.connectorHeat2DTemperature connThermoB;
 //
 parameter SIu.Length L                  "Length of the wall, m";
 parameter SIu.ThermalConductivity k = 0.1       "(Constant) Thermal conductivity, W/(m K)";
 parameter SIu.Density rho = 500             "(Constant) Density, kg/m^3";
 parameter SIu.SpecificHeatCapacity cp = 1200       "(Constant) Specific heat at constant pressure, J/(kg K)";
 parameter SIu.Temperature Ti = 293.15          "Initial temperatuere, K";
 parameter SIu.Thickness thickness = 0.05         "Thickness of wall, m";
 parameter Integer num_sL = 3               "Number of subLayers";
 parameter Integer num_nodes = num_sL + 1         "Number of nodes";
 parameter SIu.Thickness delx = thickness/num_sL      "Thickness of a subLayer";
 SIu.Temperature T[num_nodes]                "Temperature at the nodes, K" annotation(HideResult=true);
 parameter SIu.Position x[num_nodes] = {r-(thickness/2) for r in 0:delx:thickness};
initial equation
 T[:] = Ti*ones(num_nodes);
equation
 // ....................................................
 // define some things at end nodes/connectors A and B
 connThermoA.T = T[1];
```

```
 connThermoB.T = T[end];
 // ..................................................
 // heat diffusion equation (consv. of energy)
 // boundary node/connector A
 rho*cp*(delx/2)*L*der(T[1]) = connThermoA.Q2D + k*L*((T[2] - T[1])/delx);
 // nodes in the interior
 for i in 2:num_nodes-1 loop
  rho*cp*(delx)*L*der(T[i]) = k*L*((T[i-1] - T[i])/delx) + k*L*((T[i+1] - T[i])/delx);
 end for;
 // boundary node/connector B
 rho*cp*(delx/2)*L*der(T[end]) = k*L*((T[end-1] - T[end])/delx) + connThermoB.Q2D;
 //
end Conduction1D_HomoIsoWall;

model Convection
 BldgPhysics.Connectors.connectorHeat2DTemperature connThermo;
 parameter SIu.Length L               "Length of the wall, m";
 parameter SIu.CoefficientOfHeatTransfer h = 10  "(Constant) Convection coefficient, W/(m^2 K)";
 SIu.Temperature Tambient             "Ambient temperature in K";
equation
 connThermo.Q2D = h*L*(connThermo.T - Tambient);
end Convection;
```

REFERENCES

Addington, D. M. (2003). *Advanced Building Simulation*, chapter 6, pages 141–158. Spon Press.

Ansumali, S. (2004). Entropic lattice Boltzmann simulation of the flow past square cylinder. *International Journal of Modern Physics C*, 15(3):435–445.

Ansumali, S., Karlin, I. and Succi, S. (2004). Kinetic theory of turbulence modeling: smallness parameter, scaling and microscopic derivation of Smagorinsky model. *Physica A: Statistical Mechanics and its Applications*, 338(3-4):379–394.

Ansumali, S. and V. Karlin, I. (2002). Kinetic boundary conditions in the lattice Boltzmann method. *Physical Review E*, 66(2).

ANSYS (2010). FLUENT. Ansys, Inc., http://www.ansys.com/products/fluid-dynamics/fluent/.

Augenbroe, G. (1986). Research-Oriented Tools for Temperature Calculations in Buildings. In *System Simulation in Buildings: Proceeings of the Second International Conference on System Simulation in Buildings*, pages 234–255. Commission of the European Communities.

Augenbroe, G. (2003). *Advanced Building Simulation*, chapter 1, pages 4–24. Spon Press.

Axley, J. (2007). Multizone Airflow Modeling in Buildings: History and Theory. *HVAC & R Research*, 13(6):907–928.

Batteh, J. J. (2006). Integral Analysis for Thermo-Fluid Applications in Modelica. In *Modelica 2006 Proceedings*, pages 661–667.

Ben-Nakhi, A. and Mahmou, M. (2008). Conjugate turbulent natural convection in the roof enclosure of a heavy construction building during winter. *Applied Thermal Engineering*, 28(11-12):1522–1535.

Ben-Nakhi, A. and Mahmoud, M. (2007). Conjugate natural convection in the roof cavity of heavy construction building during summer. *Applied Thermal Engineering*, 27(2-3):287–298.

Bhatnagar, P., Gross, E. and Krook, M. (1954). A Model for Collision Processes in Gasses. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. *Physical Review*, 94(3):511—-525.

Borrmann, A., Wenisch, P., van Treeck, C. and Rank, E. (2006). Collaborative computational steering: Principles and application in HVAC layout. *Integrated Computer-Aided Engineering*, 13(4):361–376.

Bradley, D. and Kummërt, M. (2005). New Evolutions in TRNSYS - A Selection of Version 16 Features. In *Proceedings of Building Simulation 2005*, pages 107–114, Montréal, Canada. IBPSA.

Broadwell, J. E. (1964b). Shock Structure in a Simple Discrete Velocity Gas. *Physics of Fluids*, 7(8):1243—-1247.

Broadwell, J. E. (1964a). Study of rarefied shear flow by discrete velocity method. *Journal of Fluid Mechanics*, 19:401–414.

Brück, D., Elmqvist, H., Mattsson, S. E. and Olsson, H. (2002). Dymola for Multi-Engineering Modeling and Simulation. In *Proceedings of the Second International Modelica Conference*, pages 55–58. The Modelica Association.

Cao, N., Chen, S., Jin, S. and Mart\'inez, D. (1997). Physical Symmetry and Lattice Symmetry in the Lattice Boltzmann Method. *Physical Review E*, 55(1):R21—-R24.

Cellier, F. E. and Kofman, E. (2006). *Continuous System Simulation*. Springer.

Cercignani, C. (1988). *The Boltzmann Equation and its Applications*, volume 67 of *Applied Mathematical Sciences*. Springer-Verlag.

Chen, H., Orszag, S. A., Staroselsky, I. and Succi, S. (2004). Expanded analogy between Boltzmann kinetic theory of fluids and turbulence. *Journal of Fluid Mechanics*, 519:301–314.

Chen, Q. (2009). Ventilation Performance Prediction for Buildings: A Method Overview and Recent Applications. *Building and Environment*, 44(4):848—-858.

Chen, Q., Peng, X. and van Paassen, A. (1995). Prediction of Room Thermal Response by CFD Technique with Conjugate Heat Transfer and Radiation Models. *ASHRAE Transactions: Research*, 101(2):50—-60.

Chen, S. and Doolen, G. D. (1998). Lattice Boltzmann Method for Fluid Flows. *Annual Review of Fluid Mechanics*, 30:329–364.

Chen, S., Mart\'inez, D. and Mei, R. (1996). On Boundary Conditions in Lattice Boltzmann Methods. *Physics of Fluids*, 8(9):2527–2536.

Chikatamarla, S., Ansumali, S. and Karlin, I. (2006). Entropic Lattice Boltzmann Models for Hydrodynamics in Three Dimensions. *Physical Review Letters*, 97(1).

Clever, R. and Busse, F. (1974). Transition to Time-Dependent Convection. *Journal of Fluid Mechanics*, 65(OCT2):625–645.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C. and Pedersen, C. O. (2001). EnergyPlus: New Capabilities in a Whole-Building Energy Simulation Program. In *Proceedings of Building Simulation 2001*, pages 51–58, Rio de Janeiro, Brazil. IBPSA.

Crouse, B., Krafczyk, M., Kühner, S., Rank, E. and van Treeck, C. (2002). Indoor Air Flow Analysis Based on Lattice Boltzmann Methods. *Energy and Buildings*, 34:941–949.

Currie, I. G. (1993). *Fundamental Mechanics of Fluids*. McGraw-Hill, Inc..

Dassault Systemes (2010). Dymola. http://www.3ds.com.

Davis, G. (1983). Natural Conveciton of Air in a Square Cavity: A Bench Mark Numerical Solution. *International Journal for Numerical Methods in Fluids*, 3(3):249–264.

de la Fuente, L., Causon, D., Ingram, D., Mingham, C. and Raper, D. (2003). Towards simulating urban canyon circulations with a 2D lattice Boltzmann model. *Environmental Modelling & Software*, 18(1):71–79.

D'Humières, D. (1994). *Generalized Lattice Boltzmann Equations*, pages 450—-458. American Institute of Aeronautics and Astronautics.

D'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P. and Luo, L.-S. (2002). Multiple-relaxation-time lattice Boltzmann models in three dimensions.. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 360(1792):437–51.

D'Humières, D. and Lallemand, P. (1987). Numerical Simulations of Hydrodynamics with Lattice Gas Automata in Two Dimensions. *Complex Systems*, 1:599—-632.

Dixit, H. and Babu, V. (2006). Simulation of high Rayleigh number natural convection in a square cavity using the lattice Boltzmann method. *International Journal of Heat and Mass Transfer*, 49(3-4):727–739.

Djunaedy, E. (2005). *External Coupling Between Building Energy Simulation and Computational Fluid Dynamics*. PhD. Thesis, Technische Universiteit Eindhoven.

Djunaedy, E., Hensen, J. L. M. and Loomans, M. (2004). Selecting an Appropriate Tool for Airflow Simulation in Buildings. *Building Services Engineering Research and Technology*, 25(3):269–278.

Djunaedy, E., Hensen, J. L. M. and Loomans, M. (2005). External Coupling Between CFD and Energy Simulation: Implementation and Validation. In *ASHRAE Transactions*, pages 612–624. ASHRAE.

DOE (2008). *EnergyPlus Engineering Reference*. US Department of Energy.

D'Orazio, A., Corcione, M. and Celata, G. (2004). Application to natural convection enclosed flows of a lattice Boltzmann BGK model coupled with a general purpose thermal boundary condition. *International Journal of Thermal Sciences*, 43(6):575–586.

Eggels, J. and Somers, J. (1995). Numerical Simulation of Free Convective Flow Using the Lattice-Boltzmann Scheme. *International Journal of Heat and Fluid Flow*, 16(5):357–364.

Feustel, H. E. (1998). COMIS — An International Multizone Air-Flow and Contaminant COMIS — An International Multizone Air-Flow and Contaminant Transport Model. Technical Report, Lawrence Berkeley National Laboratory.

Filippova, O. and Hänel, D. (1998). Grid Refinement for Lattice-BGK Models. *Journal of Computational Physics*, 147:219–228.

Filippova, O. and Hanel, D. (2000). A novel lattice BGK approach for low Mach number combustion. *Journal of Computational Physics*, 158(2):139–160.

Frisch, U., Hasslacher, B. and Pomeau, Y. (1986). Lattice Gas Automata for the Navier-Stokes Equation. *Physical Review Letters*, 56(14):1505—-1508.

Fritzson, P. (2004). *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. IEEE Press.

Girimaji, S. (2007). Boltzmann Kinetic Equation for Filtered Fluid Turbulence. *Physical Review Letters*, 99(3).

Grad, H. (1949a). On the Kinetic Theory of Rarefied Gasses. *Communications on Pure and Applied Mathematics*, 2(4):331–407.

Guo, Z., Shi, B. and Wang, N. (2000). Lattice BGK model for incompressible Navier-Stokes equation. *Journal of Computational Physics*, 165(1):288–306.

Guo, Z., Shi, B. and Zheng, C. (2002). A coupled lattice BGK model for the Boussinesq equations. *International Journal for Numerical Methods in Fluids*, 39(4):325–342.

Guo, Z. and Zhao, T. (2003). Explicit Finite-Difference Lattice Boltzmann Method with Curvilinear Coordinates. *Physical Review E*, 67(6).

Guo, Z., Zheng, C. and Shi, B. (2002b). An Extrapolation Method for Boundary Conditions in Lattice Boltzmann Method. *Physics of Fluids*, 14(6):2007—-2010.

Guo, Z., Zheng, C. and Shi, B. (2002a). Non-Equilibrium Extrapolation Method for the Velocity and Pressure Boundary Conditions in the Lattice Boltzmann Method. *Chinese Physics*, 11(4):366—-374.

Guo, Z., Zheng, C., Shi, B. and Zhao, T. (2007). Thermal lattice Boltzmann equation for low Mach number flows: Decoupling model. *Physical Review E*, 75(3).

Harris, S. (1971). *An Introduction to the Theory of the Boltzmann Equation*. Holt, Rinehart and Winston, Inc., New York.

He, X., Chen, S. and Doolen, G. D. (1998). A Novel Thermal Model for the Lattice Boltzmann Method in Incompressible Limit. *Journal of Computational Physics*, 146:282–300.

He, X. and Luo, L.-S. (1997a). A priori derivation of the lattice Boltzmann equation. *Physical Review E*, 55(6):R6333—-R6336.

He, X. and Luo, L.-S. (1997b). Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation. *Physical Review E*, 56(6):6811–6817.

He, X., Luo, L.-S. and Dembo, M. (1996). Some progress in lattice Boltzmann method .1. Nonuniform mesh grids. *Journalof Computational Physics*, 129(2):357–363.

He, X., Zou, Q., Luo, L.-S. and Dembo, M. (1997). Analytic Solutions of Simple Flows and Analysis of Nonslip Boundary Conditions for the Lattice Boltzmann BGK Model. *Journal of Statistical Physics*, 87(1–2):115–136.

Hensen, J. (2003). *Advanced Building Simulation*, chapter 4, pages 87–118. Spon Press.

Inamuro, T., Yoshino, M. and Ogino, F. (1995). A Non-Slip Boundary Condition for Lattice Boltzmann Simulations. *Physics of Fluids*, 7(12):2928–2930.

Kadanoff, L. P. (1986). On Two Levels.

Kaminski, D. and Prakash, C. (1986). Conjugate Natural Convection in a Square Enclosure: Effect of Conduction in one of the Vertical Walls. *International Journal of Heat and Mass Transfer*, 29(12):1979—-1988.

Karlin, I., Ferrante, A. and Ottinger, H. (1999). Perfect entropy functions of the Lattice Boltzmann method. *Europhysics Letters*, 47(2):182–188.

Kuhner, S., Crouse, B., Rank, E., T\"{o}lke, J. and Krafczyk, M. (2004). From a product model to visualization: Simulation of indoor flows with Lattice-Boltzmann methods. *Computer-Aided Civil and Infrastructure Engineering*, 19(6):411–420.

Kuznik, F. and Rusaouen, G. (2007). Numerical Prediction of Natural Convection Occurring in Building Components: A Double-Population Lattice Boltzmann Method. *Numerical Heat Transfer, Part A: Applications*, 52(4):315–335.

Ladd, A. J. (1994b). Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation. Part 1. Theoretical Foundation. *Journal of Fluid Mechanics*, 271:285—-309.

Ladd, A. J. (1994a). Numerical Simulations of Particulate Suspensions via a Discretized Boltzmann Equation. Part 2. Numerical Results. *Journal of Fluid Mechanics*, 271:311—-339.

Lallemand, P. and Luo, L.-S. (2003). Theory of the lattice Boltzmann method: Acoustic and thermal properties in two and three dimensions. *Physical Review E*, 68(3).

LBNL and Ayers Sowell Associates, I. (2003). *SPARK 2.0 Reference Manual*. Lawrence Berkeley National Laboratory and Ayers Sowell Associates, Inc..

Lee, T. and Lin, C.-L. (2001). A characteristic Galerkin method for discrete Boltzmann equation. *Journal of Computational Physics*, 171(1):336–356.

Li, Q., He, Y. L., Wang, Y. and Tang, G. H. (2008). An improved thermal lattice Boltzmann model for flows without viscous heat dissipation and compression work. *International Journal of Modern Physics C*, 19(1):125–150.

Liboff, R. L. (1998). *Kinetic Theory*. John Wiley and Sons, Second edition.

Lienhard(IV), J. H. and Lienhard(V), J. H. (2005). *A Heat Transfer Textbook*. Phlogiston Press, Cambridge, Massachussets, 3 edition.

Megri, A. C. and Haghighat, F. (2007). Zonal Modeling for Simulating Indoor Environment for Buildings: Review, Recent Developments, and Applications. *HVAC & R Research*, 13(6):887–905.

Mei, R. and Shyy, W. (1998). On the Finite Difference-Based Lattice Boltzmann Method in Curvilinear Coordinates. *Journal of Computational Physics*, 143:426–448.

Meng, F., Wang, M. and Li, Z. (2008). Lattice Boltzmann simulations of conjugate heat transfer in high-frequency oscillating flows. *International Journal of Heat and Fluid Flow*, 29(4):1203–1210.

Mentor Graphics (2010). FloVENT. Mentor Graphics, Inc., http://www.mentor.com.

Mezrhab, A., Bouzidi, M. and Lallemand, P. (2004). Hybrid Lattice-Boltzmann Finite-Difference Simulation of Convective Flows. *Computers and Fluids*, 33:623–641.

Mezrhab, A., Jami, M., Bouzidi, M. and Lallemand, P. (2007). Analysis of radiationâĂŞnatural convection in a divided enclosure using the lattice Boltzmann method. *Computers & Fluids*, 36(2):423–434.

Mirsadeghi, M., Blocken, B. and Hensen (2009). Application of Externally-Coupled BES-CFD in HAM Engineering of the Indoor Environment. In *Proceedings of the Eleventh International IBPSA Conference*, pages 324—-331.

Mondal, B. and Mishra, S. C. (2009). The lattice Boltzmann method and the finite volume method applied to conduction-radiation problems with heat flux boundary conditions. *International Journal for Numerical Methods in Engineering*, 78(2):172–195.

Negrao, C. (1995). *Conflation of Computational Fluid Dynamics and Building Thermal Simulatin*. Ph.D Thesis, University of Strathclyde.

Noble, D. R., Chen, S., Georgiadis, J. G. and Buckius, R. O. (1995a). A Consistent Hydrodynamic Boundary Condition for the Lattice Boltzmann Method. *Physics of Fluids*, 7(1):203–209.

PELAB (2008). *OpenModelica System Documentation*. Programming Environment Laboratory, Department of Computer and Information Science, Linköping University, Sweden.

Peng, Y., Shu, C. and Chew, Y. (2003). Simplified thermal lattice Boltzmann model for incompressible thermal flows. *Physical Review E*, 68(2).

Philippi, P., Hegele, L., dos Santos, L. and Surmas, R. (2006). From the continuous to the lattice Boltzmann equation: The discretization problem and thermal models. *Physical Review E*, 73(5).

Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press.

Potter, S. and Underwood, C. (2004). A modelling method for conjugate heat transfer and fluid flow in building spaces. *Building Services Engineering Research and Technology*, 25(2):111—-125.

Prasianakis, N. and Karlin, I. (2007). Lattice Boltzmann method for thermal flow simulation on standard lattices. *Physical Review E*, 76(1).

Prasianakis, N. I. (2008). *Lattice Boltzmann Method for Thermal Compressible Flows*. Ph.D Thesis, Swiss Federal Institute of Technology.

Prasianakis, N. I., Chikatamarla, S. S., Karlin, I. V., Ansumali, S. and Boulouchos, K. (2006). Entropic lattice Boltzmann method for simulation of thermal flows. *Mathematics and Computers in Simulation*, 72:179–183.

Reichl, L. E. (1998). *A Modern Course in Statistical Physics*. John Wiley and Sons, Second edition.

Roache, P. J. (1998). *Fundamentals of Computational Fluid Dynamics*. Hermosa Publishers, Albuquerque, NM.

Rosenfeld, A. H. (1999). The Art of Energy Efficiency: Protecting the Environment with Better Technology. *Annual Review of Energy and the Environment*, 24:33–82.

Sahlin, P. (2003). On the Effects of Decoupling Airflow and Heat Balance in Building Simulation Models. *ASHRAE Transactions*, 109:788—-800.

Sahlin, P., Eriksson, L., Grozman, P., Johnsson, H. and Shapovalov, A.et al. (2004). Whole-Building Simulation with Symbolic DAE Equations and General Purpose Solvers. *Building and Environment*, 39:949–958.

Saldamli, L., Bachmann, B., Fritzson, P. and Wiesmann, H. (2005). A Framework for Describing and Solving PDE Models in Modelica. In *Proceedings of the Fourth International Modelica Conference*. The Modelica Association.

Schlichting, H. (1979). *Boundary-Layer Theory*. McGraw-Hill, Inc..

Shan, X. (1997). Simulation of Rayleigh-Bénard convection using a lattice Boltzmann method. American Physical Society.

Shan, X. and Chen, H. (2007). A general multiple-relaxation-time Boltzmann collision model. *International Journal of Modern Physics C*, 18(4):635–643.

Shan, X., Yuan, X.-F. and Chen, H. (2006). Kinetic theory representation of hydrodynamics: a way beyond the NavierâĂŞStokes equation. *Journal of Fluid Mechanics*, 550(-1):413—-441.

Shi, Y., Zhao, T. and Guo, Z. (2004). Thermal lattice Bhatnagar-Gross-Krook model for flows with viscous heat dissipation in the incompressible limit. *Physical Review E*, 70(6).

Skordos, P. (1993). Initial and Boundary Conditions for the Lattice Boltzmann Method. *Physical Review E*, 48(6):4823—-4842.

Sowell, E. F., Moshier, M. A., Haves, P. and Curtil, D. (2004). Graph–Theoretic Methods in Simulation Using SPARK. In *Proc. High Performance Computing Symposium of the Advanced Simulation Technologies Conference*. Society for Modeling Simulation International.

Sterling, J. D. and Chen, S. (1996). Stability Analysis of Lattice Boltzmann Methods. *Journal of Computational Physics*, 123:196–206.

Stiebler, M., Tolke, J. and Krafczyk, M. (2006). An upwind discretization scheme for the finite volume lattice Boltzmann method. *Computers & Fluids*, 35(8-9):814–819.

Sturge, M. D. (2003). *Statistical and Thermal Physics*. AK Peters.

Succi, S., Chen, H. and Orszag, S. (2006). Relaxation approximations and kinetic models of fluid turbulence. *Physica A: Statistical Mechanics and its Applications*, 362(1):1–5.

Succi, S., Karlin, I. V. and Chen, H. (2002). Colloquium: Role of the H Theorem in Lattice Boltzmann Hydrodynamic Simulations. *Reviews of Modern Physics*, 74:1203–1220.

Surmas, R., Pico Ortiz, C. E. and Philippi, P. C. (2009). Simulating thermohydrodynamics by finite difference solutions of the Boltzmann equation. *The European Physical Journal Special Topics*, 171(1):81–90.

Tan, G. and Glicksman, L. R. (2005). Application of Integrating Multi-zone Model with CFD Simulation to Natural Ventilation Predicition. *Energy and Buildings*, 37:1049–1057.

Tannehill, J. C., Anderson, D. A. and Pletcher, R. H. (1997). *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, Washington, DC, Second edition.

The Modelica Association (2007). *Modelica - A Unified Object-Oriented Language for Physical Systems Modeling; Language Specification Version 3.0*. The Modelica Association, www.modelica.org.

The Modelica Association (2010). Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2. The Modelica Association, www.modelica.org.

Tritton, D. J. (1997). *Physical Fluid Dynamics*. Oxford University Press.

van Treeck, C., Rank, E., Krafczyk, M., Tolke, J. and Nachtwey, B. (2006). Extension of a Hybrid Thermal LBE Scheme for Large-Eddy Simulations of Turbulent Convective Flows. *Computers and Fluids*, 35:863–871.

Walton, G. N. and Dols, W. S. (2008). *CONTAM 2.4c User Guide and Program Documentation*. National Institute of Standards and technology.

Wang, J., Wang, M. and Li, Z. (2007). A lattice Boltzmann algorithm for fluidâĂŞsolid conjugate heat transferâŸĘ. *International Journal of Thermal Sciences*, 46(3):228–234.

Wang, L. and Chen, Q. (2007a). Theoretical and Numerical Studies of Coupling Multizone and CFD Models for Building Air Distribution Simulations. *Indoor Air*, 17(5):348–361.

Wang, L. and Chen, Q. (2007b). Validation of a Coupled Multizone-CFD Program for Building Airflow and Contaminant Transport Simulations. *HVAC & R Research*, 13(2):267–281.

Wang, L. and Chen, Q. (2008). Evaluation of Some Assumptions Used in Multizone Airflow Network Models. *Building and Environment*, 43:1671–1677.

Wetter, M. (2006a). Multizone Airflow Model in Modelica. In *Modelica 2006 Proceedings*, pages 431–440.

Wetter, M. and Haugstetter, C. (2006). Modelica vs. TRNSYS - A Comparison Between an Equation-Based and a Procedural Modeling Language for Building Energy Simulation. In *Proc. of the 2nd SimBuild Conference*.

White, F. M. (2006). *Viscous Fluid Flow*. McGraw Hill, New York, 3rd edition.

Wolf-Gladrow, D. A. (2000). *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Springer, New York.

Yu, D., Mei, R., Luo, L.-S. and Shyy, W. (2003). Viscous Flow Computations with the Method of Lattice Boltzmann Equation. *Progress in Aerospace Sciences*, 39:329–367.

Yu, H. (2004). *Lattice Boltzmann Equation Simulations of Turbulence, Mixing, and Combustion*. PhD thesis, Texas A & M University.

Yu, H. and Girimaji, S. S. (2005). Near-Field Turbulent Simulations of Rectangular Jets Using Lattice Boltzmann Method. *Physics of Fluids*, 17:125106–1—-125106–17.

Zhai, Z. (2003). *Developing an Integrated Building Design Tool by Coupling Building Energy Simulation and Computational Fluid Dynamics Programs*. Ph.D Thesis, Massachusetts Institute of Technology.

Zhai, Z. and Chen, Q. (2003). Solution Characters of Iterative Coupling Between Energy Simulation and CFD Programs. *Energy and Buildings*, 35(5):493–505.

Zhai, Z. and Chen, Q. (2004). Numerical Determination and Treatment of Convective Heat Transfer Coefficient in the Coupled Building Energy and CFD Simulation. *Building and Environment*, 39:1001–1009.

Zhai, Z. and Chen, Q. (2005). Performance of Coupled Building Energy and CFD Simulations. *Energy and Buildings*, 37(4):333–344.

Zhou, Y., Zhang, R., Staroselsky, I. and Chen, H. (2004). Numerical Simulation of Laminar and Turbulent Buoyancy-Driven Flows Using a Lattice Boltzmann Based Algorithm. *International Journal of Heat and Mass Transfer*, 47:4869–4879.

Zou, Q. and He, X. (1997). On Pressure and Velocity Boundary Conditions for the Lattice Boltzmann BGK Model. *Physics of Fluids*, 9(6):1591–1598.

VITA

Jason Brown was born in 1972 in Arcadia, Florida. He received his Bachelors of Science in Engineering from Baylor University in 1995 and a Masters of Science in Mechanical Engineering from Georgia Tech in 1998. For four years he worked as a research technician in the School of Biology at Georgia Tech, building experimental apparatus and conducting experiments on the fluid-mechanical environment of marine zooplankton in support of research in sensory ecology and oceanic carbon cycling.

Seeking to broaden his work, Jason came to the College of Architecture in 2004, first as a M.Arch student, then transferring tot he Ph.D program with the interest of applying engineering to architecture. He participated in Georgia Tech's entry in the 2007 Solar Decathlon as a student leader, learning many practical things about the design, construction, and operation of such houses. During his time as a student, he also became the instructors of record for the two environmental systems classes, and started the Building Physics class as part of the High Performance Buildings Masters program. He is a member of the American Society of Mechanical Engineers and the American Society of Heating, Refrigerating and Air-Conditioning Engineers.